


Snowflake: An Adaptive Energy and Delay Efficient Scheme for Source Location Privacy in Wireless Sensor Networks

Sain Saginbekov¹ ^a and Dossay Oryspayev²

¹Computer Science Department, School of Engineering and Digital Sciences, Nazarbayev University, Nur-Sultan 010000, Kazakhstan

²Computational Science Initiative, Brookhaven National Laboratory, Upton, NY, U.S.A.

Keywords: Wireless Sensor Networks, Source Location Privacy, Global Adversary.

Abstract: Wireless Sensor Networks (WSNs) consist of a number of resource-constrained sensor nodes and a designated node called a sink, which collects data from the sensor nodes. A WSN can be used in numerous applications such as subject tracking and monitoring, where it is often desirable to keep the location of the subject private. In these types of applications, an adversary can locate the monitored subject, if a location privacy protection scheme is not applied. In this paper, we propose an adaptive energy and delay efficient scheme, called Snowflake, that conceals the location of subjects from a global adversary. Snowflake can be adapted to make the delivery delay smaller, or to make the packet overhead low. The simulation results show that Snowflake performs better than an existing algorithm.

1 INTRODUCTION

Wireless Sensor Networks (WSNs) consist of sensor nodes that are spread in some area where they communicate with one another via wireless links upon sensing the particular specified action. Each of these sensor nodes have limited energy, computational power, and storage. As such they are low-cost, and they can be installed in large numbers. One of the nodes serves as the so-called sink node, to which all other nodes in the network try to relay their information after sensing the particular activity, possibly after processing the data as well. All of the sensor nodes installed in one network do not necessarily have to be of same kind, i.e., homogeneous, in fact sensor nodes can be heterogeneous, and this heterogeneity will not cause any issues for them to communicate with one another in the networks (Akyildiz et al., 2002a). Given all of these, it's not surprising that these WSNs can be used in many different applications such as military and industrial (Akyildiz et al., 2002a).

One of the most common applications of WSNs is object monitoring and tracking, where each sensor node relays information detected regarding the object to the sink via it's closest neighbor nodes. The common example of this is the panda hunter prob-

lem introduced by (Ozturk et al., 2004), (Kamat et al., 2005). In this case, e.g., the scientists might need to study the habitat, numbers, and locations of pandas as they grow, interact, look for food, and migrate via a WSN, but at the same time these info, especially the ones regarding the locations, must be protected from hunters, who might try to capture the signals sent via sensor nodes in order to catch the pandas. So, in safety of endangered animals the privacy of the path from where the signal originates and how and via which set of nodes it's being delivered to the sink needs to be protected in order to save these endangered animals from poachers. Another area of application is military, where friendly soldiers and/or vehicles locations need to be hidden from enemies (Conti et al., 2013). Thus the information regarding the location of the object needs to be hidden, and be made almost impossible for the attacker or enemy to acquire with their tools.

Source Location Privacy (SLP) is an important problem in WSNs, and being able to provide one has been studied extensively in the literature. Many protocols were proposed, which all try to address the issues of this problem. The solutions proposed in the literature can be categorized mainly based on what type of techniques are being utilized, and what type of adversary is being assumed (Conti et al., 2013). For example, there are two types of adversaries that

^a  <https://orcid.org/0000-0003-2313-627X>

can be assumed: one is called the *global adversary*, where the adversary can hear the communication of the whole network; and the other one is called the *local adversary*, where the adversary can only hear the partial subset of the communications exchanged among subset of overall nodes.

To that end, in this paper, we develop yet another scheme that provides location privacy in WSNs against *global attackers*. As such, in this paper, we make the following contributions:

- We propose an adaptive energy and delay efficient scheme that provides source location privacy against the global attacker.
- We simulate our scheme and compare the results with an existing algorithm.

The rest of the paper is organized as follows: Section II reviews the related work. Section III presents the network and attacker model we assume in our scheme, and defines the problem we address in this work. Section IV explains the proposed scheme. In Section V, we show the simulation results. We conclude the paper in Section VI.

2 RELATED WORKS

There exist a number of solutions to the SLP problem. In general, those solutions can be grouped into two: those that provide location privacy against local adversary and those that provide location privacy against global adversary. Former solutions usually use different paths or random paths for packets to hide the source node (Kamat et al., 2005), (Wang et al., 2008) (Spachos et al., 2014), (Kumar et al., 2015), (Berdibek and Saginbekov, 2019). The techniques used in these solution are ineffective against the global adversary that can monitor the entire network. Whereas the latter solutions use fake sources and fake packets to obfuscate the global adversary (Chen and Lou, 2010), (Jhumka et al., 2011), (Mehta et al., 2007).

In (Yang et al., 2013), the authors proposed a scheme where sensor nodes transmit a packet every interval. If there is no real packet to transmit, then the node transmits a fake packet. This interval based packet transmission introduces delay. Moreover, constant fake packets transmissions incur additional energy overhead. Therefore, the same authors propose solutions that try to reduce fake packet transmissions by transmitting packets probabilistically. In their scheme, fake packets are transmitted based on a probability p . Therefore, the larger the value of p , the higher the packet overhead. In the scheme, if a node

receives a packet or has no packet to send, the node forwards or generates a fake packet, respectively, and sends that packet based on p during its transmission time. In (Ouyang et al., 2008), the authors proposed three algorithms that are based on pseudo random numbers, which define the packet transmission time of the node. One of these three algorithms uses a probability to decide whether to send a fake packet or not when the time to send a packet occurs.

Another scheme which solves the SLP problem against the global attacker is proposed in (Bushnag et al., 2017). In their protocol, called EDAD, each sensor node computes its own sending interval based on the equation

$$t = \ln(p) / -\lambda \quad (1)$$

, where p is the probability, λ is the transmission rate, and t is the next sending interval.

3 NETWORK AND ADVERSARY MODELS

In this section, we introduce the network and adversary models used in our scheme.

3.1 Network Model

A wireless sensor network is considered as a graph $G = (V, E)$, where V is the set of nodes, and E is the set of links that exist between any two nodes. A link $(u, v) \in E$, if u is in the communication range of v and vice versa, that is we assume G to be an undirected graph. We assume that the nodes and sink are static. The nodes are randomly placed in the network. We assume homogeneous nodes, that is, all nodes have the same capabilities and consume the same amount of energy for the same operation. We call a node that senses an event a *source node*. A node that generates a random packet to obfuscate the adversary is called a *fake source*. A *real packet* is a packet generated by a source node. A *fake packet* is a packet generated by a fake source. A *routing protocol* is used to choose a routing path to route sensed data from a source node to the sink. A *parent node* v of a node u is the node, $(u, v) \in E$, which is used to forward the packet of u towards the sink. We also assume that the network is event-driven, that is, when a node detects an event, it starts transmitting packets via a routing path to the sink for a certain time period.

As mentioned above, there exists mainly two types of adversaries. In the next section we present the adversary model that we assume in our work.

3.2 Adversary Model

Given the network model above, the goal of adversary is to find the location of source node. In this section we describe what type of adversary we assume.

According to (Conti et al., 2013), adversaries can be classified according to different categories: based on their behaviors, adversaries can be external or internal, and can be active or passive. An external adversary cannot compromise a sensor node whereas an internal sensor node can. An active adversary can modify the behavior or the message traffic of the nodes whereas a passive adversary cannot. More about adversary types can be found in (Conti et al., 2013).

Based on its capabilities, an adversary can be classified as global or local (Conti et al., 2013). A global adversary can view the entire network whereas a local adversary can view only the neighbourhood. More specifically, a local adversary needs to trace the transmitted packets to locate the source as the adversary can see only its neighbourhood. Whereas a global adversary need to trace packets as it can see the source directly from the message traffic. Therefore, it is harder to provide location privacy against global adversaries.

In this work, we assume a global, external, and passive adversary. In other words, we assume the adversary to have a device that is capable to monitor the whole network traffic. We also assume that the adversary knows the locations of all sensor nodes in the network, can eavesdrop, store and analyze packets. We also assume that the adversary cannot alter packets, compromise nodes, and identify the source location based on the eavesdropped packets.

4 PROBLEM STATEMENT AND PROPOSED SOLUTION

In this section, we define the problem and present a solution for the problem.

4.1 Problem Statement

Given the network and adversary models, the question is: how source nodes are able to quickly route the packets towards the sink such that the global adversary cannot detect the source nodes, and at the same time utilize the energy efficiently.

If the source node is the only node transmitting in a given time interval, as the global adversary has the ability to monitor message transmissions in the network, the adversary can locate the source node with-

out difficulty. For that reason, not only the source node, but also fake sources should keep transmitting fake packets to obfuscate the adversary. As the most energy is consumed due to packet transmissions, the number of transmitted fake packets should be reduced as much as possible without decreasing the privacy of the source nodes; and the real packets should be routed towards the sink as quick as possible. Now the question is which nodes should transmit fake packets, how frequently they should transmit, and how real packets should be routed towards the sink.

Formally, we can summarize the problem as follows: Given a network topology $G = (V, E)$, the total number of transmitted packets $P^T = P_R^T + P_F^T$ during the interval T , where P_R^T and P_F^T are the numbers of transmitted real and fake packets, respectively. As the number of real packets, P_R^T , cannot be decreased, the objective is to reduce P_F^T as much as possible and select a path $p = s, u, v, \dots, S$, where s is a source node and S is the sink, such that p delivers the real packet of s as quickly as possible.

4.2 Proposed Solution

A straightforward solution to this problem is to let all nodes, both source nodes and fake sources, transmit packets periodically even if there is no real packet to send. That is, in every time interval $t \leq T$, $P^t = P_R^t + P_F^t = |V|$ even if $P_R^t = 0$, which means $P^t = P_F^t = |V|$. In such a case, the adversary cannot differentiate the source node from fake sources. However, transmitting fake packets periodically is costly as the packet transmission and reception operations consume the largest amount of energy (Akyildiz et al., 2002b).

Another solution is to have nodes send packets probabilistically, that is, as in (Ouyang et al., 2008), (Bushnag et al., 2017), nodes send packets based on a probability. In a specific time interval, a node randomly chooses a number, if that number is smaller than a threshold value, then the node sends a packet. However, there is a trade-off between packet overhead and delay. If the delay is decreased, then the packet overhead is increased or vice versa. Moreover, this approach decreases the delivery ratio. That is, real packets may not reach the sink.

Our scheme, called Snowflake (see Algorithm 1), addresses this problem by changing the mechanism of sending fake packets. Snowflake does not use a probabilistic packet transmission, but uses a periodic packet transmissions to make the delivery ratio 100%. Despite periodic packet transmissions Snowflake can still reduce the packet overhead and delay. The idea is to have relatively small number of nodes, called *main nodes*, send packets frequently, and the rest, a con-

siderable amount of nodes, called *secondary nodes*, send packets less frequently. We believe that this approach reduces the packet overhead. However, if a source node selects a routing path consisting of only secondary nodes, then the delay may be quite large. Therefore, to decrease the delay, i.e., to deliver the real packet to the sink quicker, the routing path should also include main nodes which can take the packet to the sink quicker.

The Snowflake scheme includes two phases: the setup phase and the routing phase. In the setup phase (Algorithm 1), a shortest path tree rooted at the sink is built using the Breadth-First Search algorithm (Cormen et al., 2001). Then starting from the pre-lowest level nodes up to the second level nodes, every node transmits a control packet to its parent to inform about its number of children. When the sink receives the number of children from all of its children, it broadcasts a control packet, called a *main* packet to all its children. The main packet includes three values: i) τ , the time interval during which a node transmits one packet; ii) α , called *adapt* value, which is used by a secondary node to compute its own time interval; and iii) node id. The main packet is used to choose main paths among all possible paths started from the sink (see Figure 1). Whenever a node receives the main packet, it checks if the value in the id field corresponds to its own id, if the ids are equal, then the node indicates itself as a main node, saves τ , and broadcasts that packet to its children, including the id of the node with the largest number of children. When there are more than one children with the same number of children, then the parent node can randomly choose one of them. If the value in the id field does not correspond to the id of the receiving node, then the node saves the value of α and τ in the packet, and broadcast the main packet with $id = \emptyset$. The transmission of main packets stops when leaf nodes receive the main packet. All nodes other than main nodes become secondary nodes. After this phase the nodes will be ready to transmit (real or fake) packets. To route a packet for a node it is enough to know its parent node, τ , and α .

In the routing phase (Algorithm 2), every node transmits a packet (real or fake) every time interval. But this time interval differs depending on the node type: if it is a main node, then the node transmits every τ time interval; if it is a secondary node, then the node transmits every $\tau \times \alpha$ time interval, where α is a variable which can be adapted to change the delay and packet overhead.

The privacy level of Snowflake is similar to the one proposed in (Yang et al., 2013), as all nodes in the network constantly transmit real or fake packets.

Algorithm 1: Snowflake setup phase.

- 1: Run Breadth-First Search algorithm to build a shortest path tree rooted at the sink
 - 2: Starting from the pre-lowest up to level 2, for each node, send the number of children to its parent
 - 3: Starting from the sink, send a control packet, called a *main* packet. A node which receives the *main* packet with its *id* in it, indicates itself as a main node, otherwise, it indicates itself as a secondary node. Then the main node includes the id of its child with the largest number of children and sends the main packet to its children.
-

Algorithm 2: Snowflake routing phase.

- 1: **while true do**
 - 2: **if** Node on main road **then**
 - 3: Every τ time interval
 - 4: **if** A real packet exists **then**
 - 5: Forward the real packet to parent node
 - 6: **else**
 - 7: Send a fake packet to parent node
 - 8: **end if**
 - 9: **else**
 - 10: Every $\tau \times \alpha$ time interval
 - 11: **if** A real packet exists **then**
 - 12: Forward the real packet to parent node
 - 13: **else**
 - 14: Send a fake packet to parent node
 - 15: **end if**
 - 16: **end if**
 - 17: **end while**
-

5 SIMULATION RESULTS

We simulated Snowflake in an application written using Java programming language. We ran the simulation 1000 times in a network of 400 nodes in an area of 100 m x 100 m. The sink is placed at the center. The communication range is 15 m (see Figure 1). To assess the performance of Snowflake, we compared Snowflake with EDAD (Bushnag et al., 2017) in terms of the average packet overhead, average delivery ratio, average delay, and max delay. The packet overhead is the total number of all transmitted packets during the 100 time intervals. The delivery ratio is the ratio of the number of real packets delivered to the sink to the total number of real packets transmitted by the source nodes. The delay is the time delay between when a real packet is generated and delivered to the sink. To make the comparisons fair, we used the same time interval, τ , in both schemes. The

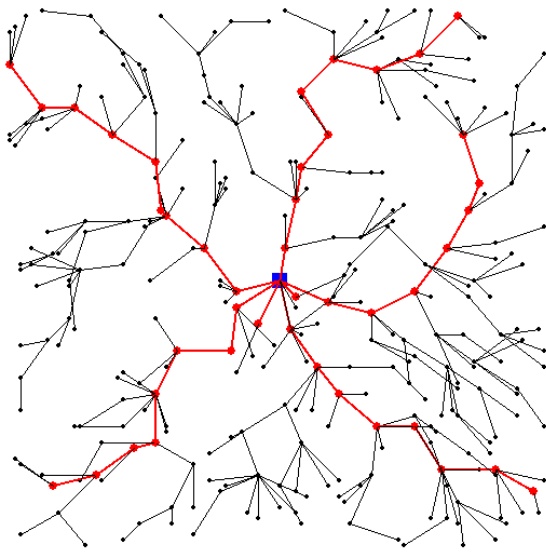


Figure 1: Network: nodes are randomly deployed. Red paths are main paths. Blue square is the sink.

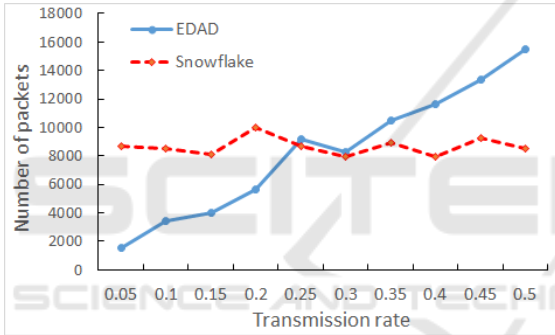


Figure 2: Packet overhead.

value of τ was generated based on the equation 1. In each of 1000 simulations, each node spends 100 time intervals. The unit of delay, t , depends on the unit of the time interval. That is, if the unit of the time interval is millisecond or second, then the unit of delay t is millisecond or second, respectively. Therefore, in the figures about delays, we used t as unit.

The results shown in Figures 2, 3, 4, and 5, are all simulated using $\alpha = 4$. Figure 2 shows the average number of packets transmitted by all nodes in 100 time intervals. In Snowflake, the number of transmitted packets does not differ much (as packet transmission does not depend on a probability), whereas in EDAD that number increases as the transmission rate increases. As can be seen from the figure, for the transmission rates of 0.25 and bigger, Snowflake shows better results.

Figures 3 and 4 show the average delay and max delay, respectively. We can see from the figures that Snowflake shows better average and max delays. Moreover, Snowflake's max delay is lower than

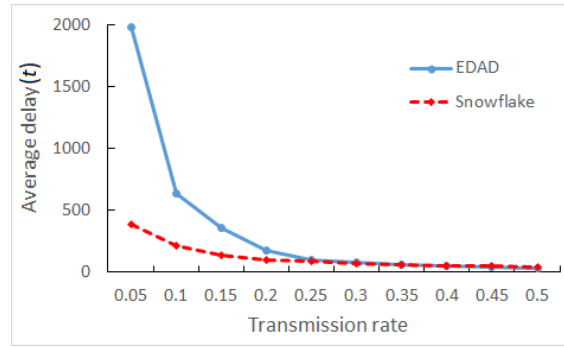


Figure 3: Average delay.

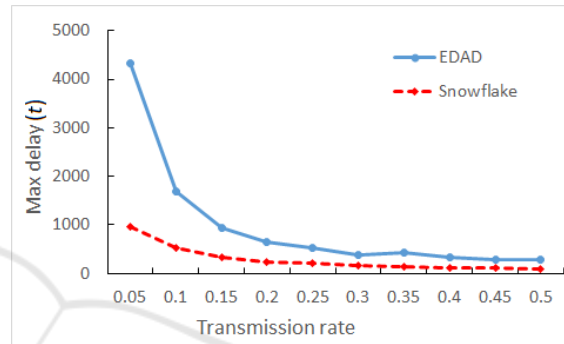


Figure 4: Max delay.

EDAD's average delay. Although Snowflake does not depend on the transmission rates, as it can be seen from the figures, for small transmission rates Snowflake shows higher delays. This is because Snowflake uses the time interval obtained from the equation given in Section 2 which outputs smaller time interval for smaller transmission rates.

In Figure 5 it can be seen that for low transmission ratios, EDAD does not provide 100% delivery whereas Snowflake has always the delivery ratio of 100%.

Figures 6 and 7 show the performance of Snowflake based on different α (adapt) values. As can be seen, Snowflake can incur lower packet overhead as we increase the value of α . Whereas the average delay is still comparable with the EDAD's average delay.

In summary, according to the simulation results, Snowflake can provide source location privacy while keeping the packet overhead and delay low.

6 CONCLUSION AND FUTURE WORK

In this paper, we have presented yet another scheme, called Snowflake, which provides source location privacy in the presence of global adversary. We made a

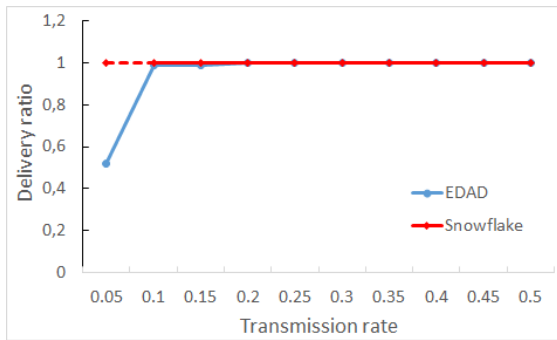


Figure 5: Delivery ratio.

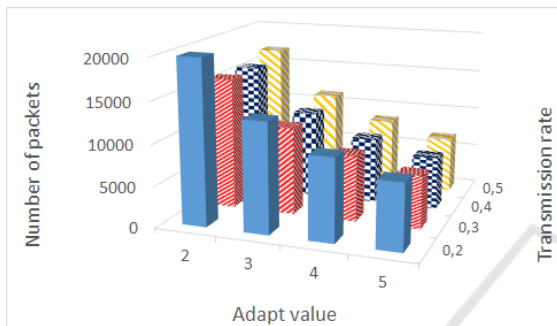


Figure 6: Snowflake: packet overhead for different α (adapt) values.

number of simulation experiments and showed that Snowflake performs well compared to an existing scheme in terms of packet overhead, delay, and delivery ratio. For future work we plan to improve the scheme and prove some theoretical results.

REFERENCES

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002a). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422.

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002b). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422.

Berdibek, A. and Saginbekov, S. (2019). A routing protocol for source location privacy in wireless sensor networks with multiple sources. *Q2SWinet'19*, New York, NY, USA. Association for Computing Machinery.

Bushnag, A., Abuzneid, A., and Mahmood, A. (2017). An efficient source anonymity technique based on exponential distribution against a global adversary model using fake injections. In *Proceedings of the 13th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Q2SWinet '17, page 15–21, New York, NY, USA. Association for Computing Machinery.

Chen, H. and Lou, W. (2010). From nowhere to somewhere: Protecting end-to-end location privacy in wireless sensor networks. pages 1–8.

Conti, M., Willemsen, J., and Crispo, B. (2013). Providing

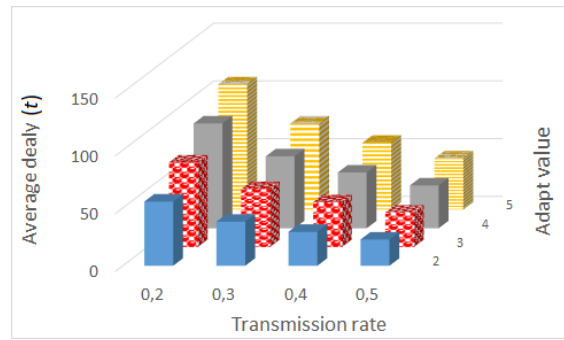


Figure 7: Snowflake: average delay for different α (adapt) values.

source location privacy in wireless sensor networks: A survey. *IEEE Communications Surveys and Tutorials*, 15(3):1238–1280.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. The MIT Press, 2nd edition.

Jhumka, A., Leeke, M., and Shrestha, S. (2011). On the use of fake sources for source location privacy. *Comput. J.*, 54(6):860–874.

Kamat, P., Zhang, Y., Trappe, W., and Ozturk, C. (2005). Enhancing source-location privacy in sensor network routing. In *25th IEEE international conference on distributed computing systems (ICDCS'05)*, pages 599–608. IEEE.

Kumar, P., Singh, J. P., Vishnoi, P., and Singh, M. P. (2015). Source location privacy using multiple-phantom nodes in wsn. In *TENCON 2015 - 2015 IEEE Region 10 Conference*, pages 1–6.

Mehta, K., Liu, D., and Wright, M. (2007). Location privacy in sensor networks against a global eavesdropper. In *Proceedings of the 15th IEEE International Conference on Network Protocols*, pages 314–323, Beijing, China.

Ouyang, Y., Le, Z., Liu, D., Ford, J., and Makedon, F. (2008). Source location privacy against laptop-class attacks in sensor networks. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, SecureComm '08, New York, NY, USA. Association for Computing Machinery.

Ozturk, C., Zhang, Y., and Trappe, W. (2004). Source-location privacy in energy-constrained sensor network routing. In *Proceedings of the 2Nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '04, pages 88–93. ACM.

Spachos, P., Toumpakaris, D., and Hatzinakos, D. (2014). Angle-based dynamic routing scheme for source location privacy in wireless sensor networks. In *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*, pages 1–5.

Wang, W., Chen, L., and Wang, J. (2008). A source-location privacy protocol in wsn based on locational angle. pages 1630–1634.

Yang, Y., Shao, M., Zhu, S., and Cao, G. (2013). Towards statistically strong source anonymity for sensor networks. *ACM Trans. Sen. Netw.*, 9(3).