ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

Energy Efficiency Using Load Balanced Clustering - Dual Data Uploading In Wireless Sensor Networks

[1] D.Iswarya, [2] G.Rajeswari
[1] Final Year PG Computer Science Student, [2] Asst. Prof of CSE
[1][2] Sree Sowdambika College of Engineering

Abstract **– In mobile wireless networks, the developing vicinity based applications have prompted the requirement for very compelling and vitality productive neighbor disclosure conventions. The Primary worry in a Wireless Sensor Network is Energy utilization. The primary issue in the system is the point at which the information is sent from hub to sink, the information will be lost because of low vitality of node.The structure utilizes dispersed load adjusted grouping and double information transferring, which is alluded to as LBC-DDU. It comprises of three-layer (I) sensor layer (ii) group head layer, and (iii) portable gatherer (called SenCar layer). At the sensor layer, a disseminated stack adjusted grouping (LBC) calculation is proposed for sensors to self-arrange themselves into bunches. At the bunch head layer, the between group transmission extend is deliberately ensured the network among the bunches. Numerous group heads inside a bunch participate with each other to perform vitality sparing between bunch correspondences.**

Index Terms—Mobile wireless network, neighbor discovery, protocol design , dual data .

INTRODUCTION

NOWADAYS, the transfer of data between neighboring nodes in mobile wireless networks has been increasingly indispensible owing to the rapid growth of diverse demands in people's everyday life. For instance, a college student may want to discuss a math problem with other students in the library using his/her tablet; a video game fan is likely to have a car race on the smartphone with other people in a Starbucks coffee shop. These motivate the appearance of proximity-based applications. Although central servers can be employed, proximity-based applications' potential can be better exploited providing the ability of discovering nearby mobile devices in one's wireless communication vicinity due to four reasons. First, users can enjoy the convenience of local neighbor discovery at any time, while the centralized service may be unavailable due to unexpected reasons. Second, a single neighbor discovery protocol can benefit various applications by providing more flexibility than the centralized approach. Third, communications between a central server and different mobile nodes may induce problems, such as excessive transmission overheads, congestion, and unexpected reaction delay. Last but not least, searching for nearby mobile

Therefore, a distributed neighbor discovery protocol for mobile wireless networks is highly needed in practice.

Generally, there are three challenges in designing such a neighbor discovery protocol.

• The first one is energy efficiency. It is known that it takes the mobile devices almost a similar amount of energy to transmit and to listen to the wireless media. Due to limited battery power, a mobile node can only periodically turn on its wireless interface with a certain duty cycle. In some applications, nodes may agree on the same duty cycle for fast neighbor discovery (symmetric case). However, mobile nodes may need to adopt different duty cycles independently, according to their remaining battery power levels (asymmetric case). Therefore, both the symmetric and asymmetric neighbor discovery should be considered.

• The second challenge is effectiveness, i.e., the neighbor discovery protocol should not only guarantee successful discovery between neighboring nodes, but also realize a short latency at the same time. On one hand, the probabilistic approach in static sensor networks does not meet this requirement because it fails to provide a worst-case discovery latency bound, and thus leads to confusion between discovery failure and nonexistence of neighbors. On the other hand, the discovery latency should be short enough, so that the users will not lose patience before finding a neighbor, and the interval when two mobile

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

nodes are within each other's communication range can be captured.

• In an ideal case, neighboring nodes can discover each other immediately if they turn to awake simultaneously upon synchronized clocks. Without a central server, the synchronization can be achieved through GPS [23]. Nevertheless, it is too energy-consuming for mobile devices. Thus, how to deal with asynchronization is the third challenge to the design of a neighbor discovery protocol.

II. PURPOSE

The detailed contributions of this work are listed as follows.Load balanced clustering - dual data uploading consists of three-layer
(i) sensor layer
(ii) cluster head layer
(iii) mobile collector (called SenCar layer).

i. sensor layer:
In the sensor layer, the distributed LBC algorithm used to self organize the sensors are into the cluster.

ii.cluster head layer:
In the cluster head layer, the sensors which have high residual energy are elected as cluster head. Each cluster consists of two cluster heads called Cluster Head Groups (CHGs). The cluster head collects the data from the sensors and it transfer the data to the sink through the neighbour cluster heads by Adhoc On-demand Multipath Distance Vector (AOMDV) routing protocol.

iii. SenCar layer:
Whenever the energy level reaches below the threshold level, SenCar starts to collects the data by selecting the polling points in the SenCar layer by the two antennas and it transfers to the sink simultaneously .

III. SYSTEM MODEL

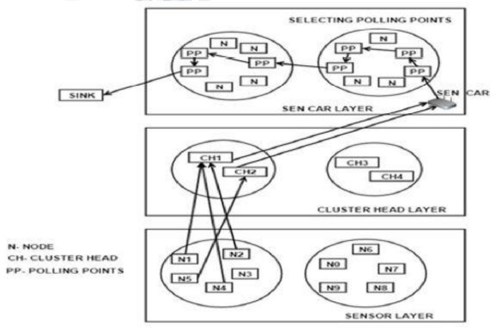


Fig.1. Overviewof the LBC-DDU framework

A .Description
The overview of LBC-DDU based mobile data collection technique is shown in figure 1.

B. Sensor Layer
In the sensor layer, a distributed load balanced clustering (LBC) algorithm is proposed for sensors to self-organize themselves into clusters. In contrast to existing clustering methods, our scheme generates multiple cluster heads in each cluster to balance the work load and facilitate dual data uploading. The sensor layer is the bottom and basic layer. For generality, we do not make any assumptions on sensor distribution or node capability, such as location-awareness. Each sensor is assumed to be able to communicate only with its neighbors, i.e., the nodes within its transmission range. During initialization, sensors are self-organized into clusters. Each sensor decides to be either a cluster head or a cluster member in a distributed manner. In the end, sensors with higher residual energy would become cluster heads and each cluster has at most M cluster heads, where M is a system parameter. For convenience, the multiple cluster heads within a cluster are called a cluster head group (CHG), with each cluster head being the peer of others. The algorithm constructs clusters such that each sensor in a cluster is 1-hop away from at least one cluster head. The benefit of such organization is that the intra-cluster aggregation is limited to a single hop. In the case that a sensor may be covered by multiple cluster heads in a CHG, it can be optionally affiliated with one cluster head for load balancing.

To avoid collisions during data aggregation, the CHG adopts time-division-multiple-access based technique to co-ordinate communications between sensor nodes. Note that only intra-cluster synchronization is needed here because data are collected via SenCar. In the case of imperfectsynchronization, some hybrid techniques to combine time division multiple access with contention-based access protocols (Carrier Sense Multiple Access) that listen to the medium before transmitting are required. For example, hybrid protocols like Z-MAC can be utilized to enhance the strengths and offset the weaknesses of TDMA and CSMA. Upon the arrival of SenCar, each CHG uploads buffered data via MU-MIMO communications and synchronizes its local clocks with the global clock on SenCar via acknowledgement messages. Finally, periodical reclustering is performed to rotate cluster heads among sensors with higher residual energy to avoid draining energy from cluster heads.

C. Cluster Head Layer

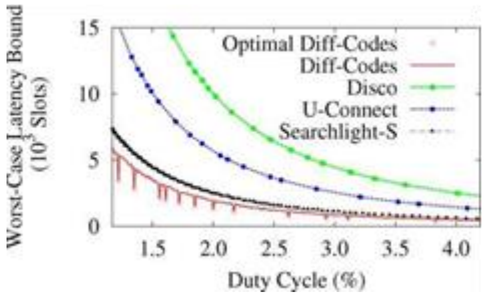
At the cluster head layer, the inter-cluster transmission range is carefully chosen to guarantee the connectivity among the clusters. Multiple cluster heads within a cluster cooperate with each other to perform energy saving intercluster communications. Through intercluster transmissions, cluster head information is forwarded to SenCar for its moving trajectory planning. The cluster head layer consists of all the cluster heads. As aforementioned, intercluster forwarding is only used to send the CHG information of each cluster to SenCar, which contains an identification list of multiple cluster heads in a CHG. Such information must be sent before SenCar departs for its data collection tour. Upon receiving this information, SenCar utilizes it to determine where to stop within each cluster to collect data from its CHG. To guarantee the connectivity for intercluster communication, the cluster heads in a CHG can cooperatively send out duplicated information to achieve spatial diversity, which provides reliable transmissions and energy saving. Moreover, cluster heads can also adjust their output power for a desirable transmission range to ensure a certain degree of connectivity among clusters.

D. Mobile Collector Layer

The mobile collector layer is SenCar it is equipped with two antennas, which enables two cluster heads to simultaneously upload data to SenCar in each time by utilizing multi-user multiple-input and multiple-output (MU-MIMO)The first step (lines 1–6) in the algorithm is to build an initial, but not necessarily feasible, code of the target length .The active slots in are determined by the optimal Diff-Code whose length is the largest among all the optimal DiffCodes shorter than . An intuitive method of initializing is to assign slot active as long as slot is active in However, we notice that for , such that the th and th slots are active in both and , if , code and will both be feasible under the slot offset of 10.
To collect data as fast as possible, SenCar should stop at positions inside a cluster that can achieve maximum capacity. In theory, since SenCar is mobile, it has the freedom to choose any preferred position. However, this is infeasible in practice, because it is very hard to estimate channel conditions for all possible positions. To mitigate the impact from dynamic channel conditions, SenCar measures channel state information before each data collection tour to select candidate locations for data collection. We call these possible locations SenCar can stop to perform concurrent data collections polling points. In fact, SenCar does not have to visit all the polling points. Instead, it calculates some polling points which are accessible and we call them selected polling points. In addition, we need to determine the sequence for SenCar to visit these selected polling points such that data collection latency is minimized. Since SenCar has pre-knowledge about the locations of polling points, it can find a good trajectory by seeking the shortest route that visits each selected polling point exactly once and then returns to the data sink.The proposed framework aims to achieve great energy saving and shortened data collection latency, which has the potential for different types of data services. Although traditional designs of WSNs can support low-rate data services, more and more sensing applications nowadays require high-definition pictures and audio/video recording, which has become an overwhelming trend for next generation sensor designs Using MU-MIMO can greatly speed up data collection time and reduce the overall latency. The application scenario emerges in disaster rescue. For example, to combat forest fire, sensor nodes are usually deployed densely to monitor the situation. These applications usually involve hundreds of readings in a short period a large amount of data and are risky for human being to manually collect sensed data. A mobile collector equipped with multiple antennas overcomes these difficulties by reducing data collection latency and reaching hazard regions not accessible by human being. Although employing mobility may elongate the moving time, data collection time would become dominant or at least comparable to moving time for many high-rate or densely deployed sensing applications. In addition, using the mobile data collector can successfully obtain data even from disconnected regions and guarantee that all of the generated data are collected.

V. EVALUATION

We not only conducted comprehensive simulations, but also prototyped our designs on a USRP-N210 testbed, to evaluate



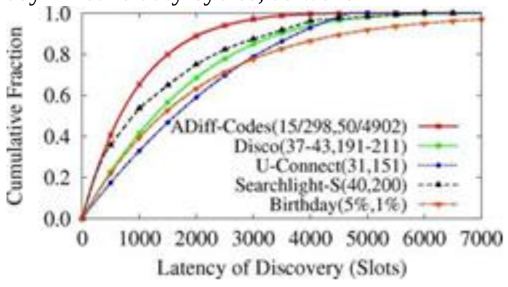
ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

Worst-case latency bound versus duty cycle.he discovery latencies with various specific symmetric and asymmetric pattern codes. For comparison, we used deterministic protocols, including Disco, U-Connect, and Searchlight-S, and a probabilistic protocol, Birthday. In this section, we first present how the worst-case latency bound changes with the symmetric duty cycle for various deterministic neighbor discovery protocols. Then, we compare both symmetric and asymmetric discovery latencies of different neighbor discovery protocols in two scenarios: one-to-one neighbor discovery and clique neighbor discovery. We compare different neighbor discovery protocols using similar duty cycles as in previous works (e.g., Disco, U-Connect , and Searchlight).

A. Worst-Case Bound of Symmetric Discovery Latency Worst-case latency bound versus duty cycle demonstrates the

worst-case latency bound of various neighbor discovery protocols restricted by symmetric duty cycle. Note that there may exist more than one pattern yielding the same duty cycle for Disco and Diff-Codes. We use adjacent prime numbers to generate Disco patterns, in which case Disco achieves better symmetric-case performance. As for Diff-Codes, we select the pattern with the smallest worst-case bound regarding each duty cycle. We observe that Diff-Codes achieve tremendously tighter worst-case latency bounds compared to the other protocols. Reduction of worst-case latency bound shows the improvements of worst-case latency bound achieved by Diff-Codes compared to the other protocols. We note that we compare Diff-Codes to other neighbor discovery protocols under exactly the same duty cycles. Compared to Searchlight-S, with the same symmetric duty cycle, Diff-Codes can lower the worst-case latency bound by more than 20% in most cases, and the maximum reduction is as high as 50 %. Specifically, those cases of maximum reduction in correspond to optimal Diff-Codes, which are in correspondence to Table I, as well. The average worst-case latency bound we compare multiple setups of ADiff-Codes, which have the same set of asymmetric duty cycles, as well.



CDF of one-to-one discovery latencies for asymmetric duty cycles 5 %–1% shows the evaluation results for various neighbor discovery protocols. The simulated ADiff-Codes outperform Searchlight-S and Disco all along. Specifically, ADiff-Codes CDF of one-to-one discovery latencies for symmetric duty cycle 5%. reductions of Diff-Codes over Searchlight-S, U-Connect, and Disco are 23.9%, 65.7%, and 80.8%, respectively. The above numerical results verify the effectiveness of Diff-Codes.

B. One-to-One Neighbor Discovery Latencies

1) Discovery Latencies in Symmetric Case: In this set of simulations, we set the duty cycle at 5% and compare the performance of two different Diff-Codes to existing protocols. We set the cycle lengths of the two Diff-Codes at 280 and 320, the pair of primes in Disco at (37, 43), the prime of U-Connect at 31, the probing period of Searchlight-S at 40 slots, and the active probability of Birthday protocol at 5%. From the cumulative distribution of discovery latencies, we can see that Diff-Codes perform the best in both the median case and worst case. Specifically, both of the two evaluated Diff-Codes realize a median gain of nearly 30% over Searchlight-S; the minimum worst-case latency of Diff-Codes is 280 slots, which is also 30% less than that of Searchlight-S.

2) Discovery Latencies in Asymmetric Case: In the simulations for asymmetric one-to-one neighbor discovery, we consider the asymmetric duty cycles of 5% and 1%. The reduce the median discovery latency by 26.0% compared to Searchlight-S, and by 47.9% compared to Disco. The worst-case gains of ADiff-Codes are 15.9% and 28.2% over Searchlight-S and Disco, respectively. Moreover, in comparison to U-Connect, ADiff-Codes reduce the median case discovery latency by as high as 59.3% and achieve smaller latencies for more than 99.9% of times, while having a worst-case bound that is only 7.1% larger.

In there is only one single series of ADiff-Codes. However, for the same set of asymmetric duty cycles, many ADiffCodes can be constructed. Hence, we compare the discovery latencies of various ADiff-Codes setups conforming to the same asymmetric duty cycles of 5% and 1%. There are seven ADiff-Codes series presented in the figure. Even though their worst-case latencies show big differences, the latencies in the median and the average cases are fairly close. That indicates ADiff-Codes can achieve a relatively stable discovery latency, despite the combination of duty cycles.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

Furthermore, from the perspective of practical implementation, it tends to be acceptable in mobile wireless networks as long as enough neighbors are found. For example, even though there may be more than 10 mobile device users in a coffee bar, a guest is satisfied to discover only four of them to start a card game. It is different from neighbor discovery in sensor networks, where failing to discover a neighbor can result in unreachable nodes in transmission. This means that if a neighbor discovery protocol has short latencies in most cases, it can satisfy a user's demands well most of the times in practice. Considering the tremendous latency gain in the median case, and the superior performance in most of the times, i.e., all along in symmetric case, and more than 99% according to the one-to-one asymmetric simulation results, our design of (A)Diff-Codes should have great advantage in reality.

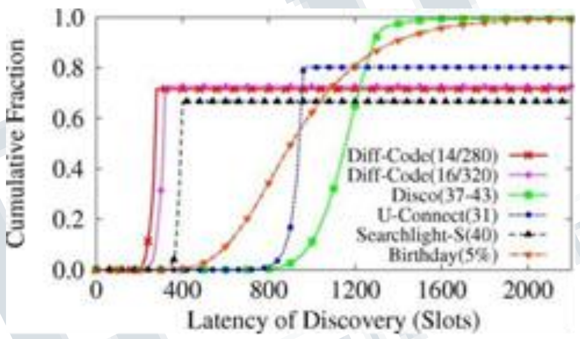
C. Clique Neighbor Discovery Latencies

Besides the one-to-one discovery latencies, we also examine the performance of our design, when there are multiple neighbors within a node's transmission range. The discovery latency in the scenario of clique neighbor discovery is the number of slots for a node to discover all its neighbors. Moreover, as explained in Section VII-B, practical applications can be satisfied by discovering enough number of neighbors. Thus, we also compare different neighbor discovery protocols using 90%-latency, i.e., the latency of discovering 90% of neighbors.

1) Discovery Latencies in Symmetric Case: For symmetric neighbor discovery with the existence of multiple neighbors, we set the duty cycle of all the nodes at 5% presents the CDF of discovery latencies when a node has 50 neighbors overall. The two simulated Diff-Codes outperform other neighbor discovery protocols significantly. Specifically, the two Diff-Codes achieve median gains of 30.7% and 20.8% compared to Searchlight-S, respectively. However, in the worst case, the evaluated neighbor discovery protocols cannot always discover all the 50 neighbors. For example, the two Diff-Codes can discover all the neighbors in 71.6% and 72.7% simulations, respectively, and Searchlight-S only converges at 66.7%. This is because the interference of concurrent discovery signals cannot be ignored in the clique scenario. Therefore, as stated in Section VII-B, we focus on the 90% latency in the following simulations of clique neighbor discovery.

To obtain how the cumulative distributions of 90% latencies change as the number of neighbors increases, we compare the CDFs of Diff-Code(14/280) to Searchlight-S, which is the best among existing protocols. As shown in, Diff-Code has better performance than Searchlight-S in all cases. The figures also indicate that both neighbor discover protocols can discover at least 90% of all the neighbors, regardless of the number of neighbors. Thus, although there are interferences among nodes in clique neighbor discovery, the user can still discover enough neighbors with high possibility.

2) Discovery Latencies in Asymmetric Case: Because the duty cycle of a mobile device is independently determined by



CDF of 90% latency with up to 200 neighbors for symmetric duty cycle 5%.

(a) Diff-Code(14/280). (b) Searchlight-S (40).

its energy budget, there may exist various duty-cycled neighbors in a node's proximity. That is to say, the node may have both symmetric neighbors with the same duty cycle and asymmetric neighbors that have different duty cycles. Hence, in our simulations, we set the duty cycle of node to be 5% and 1 %, respectively, and node has 20 neighboring nodes. For each duty cycle of node , we consider two cases: 1) all of the 20 nodes are asymmetric neighbors of node , and 2) half of the neighbors have duty cycle of 1%, and the other half have 5 % presents the CDFs of 90% latency in different cases. We note that all the deterministic neighbor discovery protocols adopt the same active-sleep patterns as in the one-to-one case, with duty cycles of 5% and 1%. The performance of Disco and U-Connect is better than that in the one-to-one case. This is because in our setups for clique neighbor discovery, most neighbors have nonaligned time-slots with node . Moreover, it is apparent

VI. CONCLUSION

In this paper, we have presented a systematic study of designing highly effective and energy-efficient neighbor-discovery protocols in mobile wireless networks. We have designed Diff-Codes for the case of symmetric duty cycle and extended it to ADiff-Codes to deal with the asymmetric case. We have derived a tighter lower bound for the worst-case latency by exploiting active slot nonalignment. Both our simulation and experiment results have shown that (A)Diff-Codes can achieve significantly better performance in both one-to-one and clique neighbor discovery, compared to state-of-art neighbor discovery protocols. Specifically, in the one-to-one scenario, Diff-Codes can reduce the worst-case latency by up to 50 % and achieve a median gain of around 30%; while ADiff-Codes are also 30% better in the median case and outperform existing neighbor discovery protocols in more than 99% simulations and experiments. In the clique scenario, both Diff-Codes and ADiff-Codes have smaller latencies to discover 90% of all the neighbors.

ACKNOWLEDGMENT

This paper was supported by the Sree Sowdambika College of Engineering, Final Year PG Computer Science student D.Iswarya (Reg.no:921816405004) guided by Asst.Prof of Computer Science Mrs. G.Rajeswari. The authors thank to their colleagues for their help and support at different stages of the system development. Finally, we would like to thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] Sony, “Sony PS Vita–Near,” [Online]. Available: http://us.playstation. com/psvita

[2] Y. Agarwal et al., “Wireless wakeups revisited: Energy management for VoIP over Wi-Fi smartphones,” in Proc. MobiSys, 2007, pp. 179–191.

[3] M. Bakht, M. Trower, and R. H. Kravets, “Searchlight: Won't you be my neighbor?,” in Proc. MobiCom, 2012, pp. 185–196.

[5] L.D.Baumert, Cyclic Difference Sets. NewYork, NY,USA:
[6] S. Bitan and T. Etzion, “Constructions for optimal constant weight cyclically permutable codes and difference families,” IEEE Trans. Inf. Theory, vol. 41, no. 1, pp. 77–87, Jan. 1995.

[7] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge, U.K.: Cambridge University Press, 2004.

[8] F. R. K. Chung, J. A. Salehi, and V. K. Wei, “Optical orthogonal codes: Design, analysis, and applications,” IEEE Trans. Inf. Theory, vol. 35 , no. 3, pp. 595–604, May 1989.

[9] P. Dutta and D. E. Culler, “Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications,” in Proc. SenSys, 2008, pp. 71–84.

[10] T. Evans and H. Mann, “On simple difference sets,” Sankhyâ, Indian J.
Statist., vol. 11, pp. 357–364, 1951.

[11] E. Felemban et al., “SAND: Sectored-antenna neighbor discovery protocol for wireless networks,” in Proc. IEEE SECON, 2010, pp. 1–9.

[12] H. Han, Y. Liu, G. Shen, Y. Zhang, and Q. Li, “DozyAP: Power-efficient Wi-Fi tethering,” in Proc. MobySys, 2012, pp. 421–434.

[13] G. G. H. Hardy and E. M. Wright, An Introduction to the Theory of Numbers. Oxford, U.K.: Oxford Univ. Press, 1979.

[14] J.-R. Jiang, Y.-C. Tseng, C.-S. Hsu, and T.-H. Lai, “Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc networks,”
Mobile Netw. Appl., vol. 10, no. 1–2, pp. 169–181, 2005.

[15] A. Kandhalu, K. Lakshmanan, and R. Rajkumar, “U-connect: A lowlatency energy-efficient asynchronous neighbor discovery protocol,” in Proc. IPSN, 2010, pp. 350–361.

[16] N. Karowski, A. C. Viana, and A. Wolisz, “Optimized asynchronous multi-channel neighbor discovery,” in Proc. IEEE INFOCOM, 2011, pp. 536–540.