2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS) | 978-1-6654-9512-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/DCOSS54816.2022.00026

Divide, Conquer and Merge for Internet-of-Things

Jagnyashini Debadarshini
IIT Bhubaneswar, jd12@iitbbs.ac.in

Sudipta Saha
IIT Bhubaneswar, sudipta@iitbbs.ac.in

***Abstract*—Divide, Conquer and Merge (DCM) is a well-known algorithmic paradigm. It has been used to solve both core problems as well as to address the scalability issue in other algorithms. It is widely used in networks or distributed computation to solve the scalability problem arising due to highly dense and wide structures. However, from the implementation point of view, all these application of DCM were designed considering the traditional *Asynchronous-Transmission* (AT) based communication technologies. In this work, we study the application of DCM in the context of *Synchronous-Transmission* (ST) which bears fundamentally different dynamics compared to the AT. Specifically, we study the application of DCM over an existing state-of-the-art ST-based strategy MiniCast and demonstrate the performance gain in calculation of aggregation. Our work, shows that applying DCM on MiniCast makes MiniCast based aggregation mechanism upto six times faster and make it consume upto five times lesser energy compared to the case when DCM is not applied.**

***Keywords*—** Asynchronous Transmission, Synchronous Transmission, Capture Effect, CI

I. Introduction

Divide Conquer and Merge (DCM) is a very well known algorithmic paradigm which approaches to solve complex problems by dividing it into two or more simpler sub-problems, conquering the sub-problems separately and combining/merging the solutions to get the final outcome. DCM has been applied in many cases either as a core solution, e.g., in Binary Search, Quick Sort, Merge Sort etc., or as a tool to improve the scalability of an existing algorithm through appropriate division . While the former application of DCM is common in algorithm development[1], the later one is more common in the context of networks and decentralized systems [2], [3], [4].

Efficient solution based on a network algorithm largely depends on the underlying communication protocols and their realization too. Compared to the traditional *Asynchronous-Transmission* (AT) based communication protocols, many recent works on *Synchronous-Transmission* (ST), have successfully shown their ability to efficiently manage communication among many more number of devices [5]. However, when a theoretical concept or a protocol is employed to serve some practical need, it always becomes more challenging than expected. For example, while covering an area of few square kilometers using an IoT-Edge, e.g., in a smart-city context, the resulting network of the IoT-devices would have a really large diameter. In these situations, a straightforward use of even the most efficient ST based protocol also may face difficulty in exhibiting sustained real-time performance.

DCM would be always a very helpful and obvious way to solve such problems. Note that in all the prior application of DCM, the underlying communication technology has been considered to be AT. Since ST and AT bear drastically different dynamics, in this work we take an endeavor to understand how efficiently DCM can be employed when the underlying communication is governed by ST. Many-to-many data sharing/aggregation being a very important and the most complex form of communication, in this work we focus on how DCM can further improve the performance of the corresponding ST based protocols.

To be specific, in this work we consider a state-of-the-art ST based many-to-many data sharing protocol *MiniCast* [6] and study how DCM can improve its performance in carrying out system-wide data-aggregation. In the following we provide a brief design of the base strategy and the key idea of applying DCM over MiniCast. Subsequently, a brief evaluation and comparison based study are provided.

II. Background

A network/distributed algorithm may face scalability issue due to high density of nodes which naturally accelerates the chance of collision among multiple packets from different source nodes. Here, application of DCM can reduce the density of the network by creating spatially co-located sparser groups and let the groups work in parallel in different frequencies/channels. Scalability issue may also originate due to high diameter of the network, e.g., while covering a large area by an IoT-edge system. This may result in significant increase in the convergence time by the underlying networking protocol [7], [8] and make it unsuitable for real-time operation. In this case, DCM can be used to divide the network into multiple spatially small disjoint groups and allow these groups to run independently in-parallel.

Spatial division of a large network into groups have been a common step to deal with scalability in several existing works. For instance, the pioneering work LEACH [9] shows a two step process where in the first step, the target task is executed locally in each of the groups whereas in the second step, the group leaders communicate the computed values directly to the sink node. In variants of the protocol, inter-group-leader communication has been also used to carry out some system-wide computation. Such two-step organization allows the non-leader nodes to turn their radio OFF when the group leaders carry out their tasks in the second step. This enables a lot of savings of power in the devices and also help to solve the

2325-2944/22/$31.00 ©2022 IEEE
DOI 10.1109/DCOSS54816.2022.00026

scalability problem especially when the system is required to cover a large area.

Here we note certain issues with the approach proposed by LEACH and its followers. The group leaders are supposed to communicate either directly to the sink node for a many-to-one operation (e.g., data collection) or execute an inter-group-leader coordination to accomplish certain system-wide goals under many-to-many model (e.g., system-wide averaging of temperature). However, in both the cases the group leaders are supposed to use long range communication so that they can talk to the sink node directly or with the other leader nodes without the help of the non-leader nodes.

The strategy described above fits quite well with certain heterogeneous settings, where different nodes have different resource capability, especially where the group leaders are fixed and equipped with more resources in the system. However, in most of the cases the system is homogeneous, i.e., composed of nodes having same type of constraints, e.g., a large scale IoT-edge composed of resource-constrained tiny devices. The basic issue in such system is to get a way to carry out inter-leader communication bypassing the other non-leader nodes. Moreover, dynamic election of the leaders keeping in mind about fairness, inter-leader/sink connectivity, energy-status, is always a substantial overhead in this model.

Fundamentally underlying communication mechanism, heavily affects the way the algorithms or strategies are developed. In that sense, LEACH and its follower protocols do their best considering an AT based communication model. Because of uncoordinated transmission, the more the number of active nodes in an AT based protocol, the more the chance of collision among the data packets which makes a solution chaotic and inefficient. By making a significant fraction of the nodes OFF in the second phase, LEACH based solutions bring a significant improvement in the performance of the system. However this specific step also makes it unsuitable for a homogeneous low-power setting.

In the context of ST, the basic dynamics is absolutely different compared to AT. *In AT, the nodes compete with each other to accomplish a job while in ST they inherently cooperate.* Thus, in AT where more number of nodes introduce a chaotic situation, in ST, in contrast, they can be exploited to make things faster, and hence better. With this point of view, we rethink the application of DCM in the context of ST and try to make the solution simpler and suitable for pure low-power homogeneous systems.

III. Design

There have been a series of systems research that deal with the many-to-many data-sharing/aggregation problem using ST [6], [7], [10]. These protocols are mainly based on the pioneering protocol Glossy [11]. It is the first protocol which successfully exploits ST for a network-wide one-to-many data sharing. The process of Glossy starts from a single node (initiator) with the broadcast of the data packet to its neighbors. The neighboring nodes in turn broadcasts the packets to their neighbors. However, in contrast to an AT based solution, Glossy organizes the process in a way so that the transmissions containing the same data packets overlap perfectly with each other which results in physical layer phenomena *constructive-interference* or *capture-effect* and enable correct reception of the packets. The process continues until the flood cover the full network.

The protocol MiniCast achieves many-to-many data sharing by enabling multiple Glossy floods from different source nodes to take place in an interspersed manner while maintaining a minimal separation among the floods with the help of fine-grained TDMA schedule. To allow all the source nodes to carry out flooding more or less at the same time it divides a transmission-slot into multiple sub-slots. Each of the nodes in the system is given a chance to transmit in one of these sub-slots. A unit of transmission where all the nodes are given chance in a sequence, is referred to as a *chain*. MiniCast takes help of TDMA, as the protocol LWB [7] does. However, unlike LWB, MiniCast enables the most compact execution of multiple different Glossy floods from different source nodes.

MiniCast originally targets many-to-many data sharing. In this work we study the application of DCM over MiniCast in the context of system-wide data aggregation tasks which cannot be done without the sharing of the actual data by each node. Note that the protocol Chaos [10] also does data aggregation in a very efficient way. However, its functionality is limited to only those aggregation functions that are not duplicate sensitive e.g. minimum, maximum [7], [12]. Unfortunately therefore, many serious data aggregation jobs such as summing, averaging, counting, derivation of distribution etc., cannot be carried out using Chaos. However, a data aggregation strategy that is based on pure data-sharing (e.g., by MiniCast) can be used for any aggregation function.

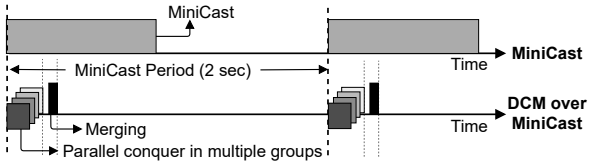

Fig. 1: Timing diagram of MiniCast and DCM over MiniCast

Application of DCM makes quite a good sense in this context as while computing such aggregation functions there is absolutely no requirement of sharing of data from every node to every other node. Rather the job can be done in groups and merged later. In this work we study DCM over MiniCast. Figure 1 shows the overall process of DCM over MiniCast as compared to the original MiniCast protocol. *Division* and *conquer* are the first two steps in DCM. Here we assume that the *division* of a system into multiple groups is assumed to be already done and *conquer*, i.e., execution of the protocols in the formed groups is shown to be happening in parallel manner. Execution of these two steps have also been studied quite well in the past works, in a scattered way though. For example, the works [13], [14] study simultaneous intra-group dissemination by using least channels/frequencies. However, note that the third step in DCM, i.e., *merging*, although

happens to be the base of the whole process, have not been explicitly addressed.

Unlike the approach taken by LEACH and its followers, use of ST allows us to take a very simple approach to achieve the merging operation. In particular, as soon as the conquer part gets over, we start running the same MiniCast protocol globally from the original initiator. To take the advantage of the *conquer* step, we reduce the sub-slots in the chain used for merging. Specifically, we make the size of the chain same as the number of groups formed. This fundamental change drastically brings down the time to converge because of two reasons. *First*, the slot-time reduces by N/d times where N is the total number of nodes and d is the number of groups. *Second*, in a single sub-slot multiple nodes try to contribute with the same value which enables the merging process converge extremely fast. This is owing to the use of ST where node fundamentally cooperate unlike AT.

IV. Evaluation

We implement the proposed DCM strategy over MiniCast in Contiki operating system for TelosB device. We run a series of experiments in the Cooja simulator with a 50 node random network spread over an area of 2500x2500 square meter. For experimenting with the DCM strategy we first divide the network into different number fixed groups having almost equal sizes. In the conquer part the base protocol is executed in different channel in different groups in parallel. The chain size in the conquer step is adjusted as per the size of the groups. In the merging step the partial results available in each of the groups (i.e., after conquering) are merged appropriately to obtain the final result. The merging operation will depend on exactly what aggregation is being computed. In most of the cases like averaging, counting, min, max etc. it will be straightforward, i.e., the final result is the reapplication of the same function over the results obtained from the individual groups.

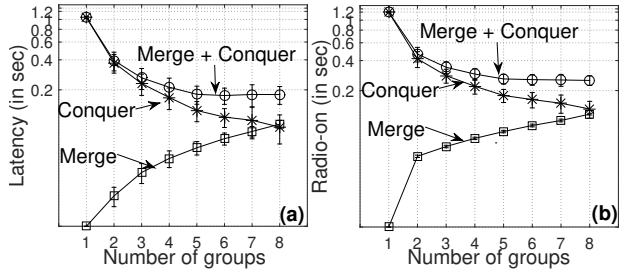

Fig. 2: Part (a) shows the latency for conquer, merge, both conquer and merge. Part (b) shows the radio-on for conquer, merge , both conquer and merge for DCM over MiniCast

An experiment with a specific group setting, is repeated for at least 1000 iterations. It has been seen that the reliability of the overall process remains consistently high (above 99%) even with the application of DCM with different group setting. Figure 2 shows the plot of the average value of the two standard metrics *latency*, i.e., the time to compute the aggregation function in each node as well as *radio-on* time, i.e., the total time for which the radio had to be kept on in each device for the DCM based execution under different group setting. The latency and the radio-on time needed for the conquer and the merge step have been also shown separately along with their combined values. It can be observed that with the increase in the number of groups, while conquer step becomes more efficient, the merging step shows the opposite behavior as it brings more overhead to merge more values coming from more number of groups. Thus, we find that beyond number of groups 5 there is almost no change in latency in this particular setting. In summary, it can be seen from the results that application of DCM can make a MiniCast based aggregation mechanism upto six times faster and make it consume upto five times lesser energy compared to the case when DCM is not applied.

V. Conclusion

DCM, i.e., *Divide-Conquer and Merge* has been one of the classical strategies for solving problems. Many networks and decentralized algorithms and protocols have taken the advantage of this paradigm for obtaining efficient solutions to complex problems. In this work we show that application of DCM can be made even more simpler in the context of ST, i.e., *Synchronous-Transmission* based communication protocols compared to the traditional AT, i.e., *Asynchronous-Transmission* based protocols. In particular we study the application of DCM over a state-of-the-art ST based many-to-many data-sharing/aggregation protocol MiniCast. Through extensive simulation based experiments we demonstrate that application of DCM can bring an enormous improvement in data sharing based system-wide aggregation operation. We also show that the proposed strategy is quite suitable for homogeneous low-power network setting.

References

[1] Tang, P., et al. "Parallelizing the merge sorting network algorithm on a multi-core computer using Go and Cilk." *ACM-SE, 2011.*

[2] Caria, M., et al. "Divide and conquer: Partitioning OSPF networks with SDN." *IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015.*

[3] Shiu, L. C., et al. "The divide-and-conquer deployment algorithm based on triangles for wireless sensor networks." *IEEE Sensors Journal, 2010.*

[4] Darties, B., et al. "A divide-and-conquer scheme for assigning roles in multi-channel wireless mesh networks." *IEEE LCN, 2009.*

[5] Zimmerling, M., et al. "Synchronous transmissions in low-power wireless: A survey of communication protocols and network services." *ACM CSUR, 2020.*

[6] Saha, S., et al. "Efficient many-to-many data sharing using synchronous transmission and TDMA." *IEEE DCOSS, 2017.*

[7] Ferrari, F., et al. "Low-power wireless bus." *ACM SenSys, 2012.*

[8] Sutton, F., et al. "The design of a responsive and energy-efficient event-triggered wireless sensing system." *IEEE EWSN, 2017.*

[9] Singh, S. K., et al. "A survey on successors of LEACH protocol." *IEEE Access, 2017.*

[10] Landsiedel, O., et al. "Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale." *ACM SenSys, 2013.*

[11] Ferrari, F., et al. "Efficient network flooding and time synchronization with glossy." *ACM/IEEE IPSN, 2011.*

[12] Nath, S., et al. "Synopsis diffusion for robust aggregation in sensor networks." *ACM TOSN, 2008.*

[13] Debadarshini, J., et al. "Fine-grained Frequencies for Simultaneous Intra-Group One-to-All Dissemination." *IEEE MASS, 2020.*

[14] Debadarshini, J., et al. "Start of Frame Delimiters (SFDs) for Simultaneous Intra-Group One-to-All Dissemination." *IEEE LCN, 2020.*