FlexiCast: Fast Data Sharing in Skewed Wireless Sensor Networks

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

02-02-2022 / 10-02-2022

CITATION

Tummala, Madhav; Saha, Sudipta; Bharadwaj, Sourabha (2022): FlexiCast: Fast Data Sharing in Skewed Wireless Sensor Networks. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.19105589.v1>

DOI

[10.36227/techrxiv.19105589.v1](https://doi.org/10.36227/techrxiv.19105589.v1)

FlexiCast: Fast Data Sharing in Skewed Wireless Sensor Networks

Madhav Tummala, Sudipta Saha, and Sourabha Bharadwaj

Abstract

Efficient sharing of data is a key component in any decentralized system. There have been quite significant developments in the data-sharing protocols, especially with the advent of the *synchronous/concurrent transmission* based strategies. However, most of these existing data sharing strategies are tested either in small scale simulated networks or in testbed settings with fixed usual homogeneous structures. Real-life IoT/WSN networks, in contrast, often bear diverse structural patterns due to a highly skewed distribution of the nodes, e.g., highly dense in some places, while sparsely distributed along pathways in some other places. Our study reveals that while the existing state-of-the-art protocols perform very well in uniformly distributed networks, they dwindle in face of these skewed networks commonly found in real life. Lack of the ability of these protocols to self-adjust as per the underlying network structure is one of the prime reasons for such performance degradation. In this work, we design and implement a flexible and self-adjusting many-to-many communication protocol *FlexiCast* and demonstrate that it consistently performs better than the current state-of-the-art protocols in a wide variety of networks. We show that FlexiCast can achieve all-to-all data sharing up to 1.77 times faster while consuming up to 1.87 times lesser energy compared to the state-of-the-art protocols.

Index Terms

Many-to-Many data sharing, WSN, IoT, Self-adjusting protocol, Concurrent-Transmission, Capture effect, TDMA, Time-varying schedule.

Madhav Tummala is currently associated with Goldman Sachs. Sudipta Saha is an Assistant Professor in the School of Electrical Sciences, IIT Bhubaneswar. Sourabha Bharadwaj is currently associated with D.E. Shaw India Private Limited. Portions of this research work were done while Madhav Tummala and Sourabha Bharadwaj were students in the School of Electrical Sciences, IIT Bhubaneswar. The authors would like to thank IIT Bhubaneswar for the financial support.

I. INTRODUCTION

Design and deployment of real-life large decentralized systems often face various practical challenges and constraints. For example, deployment of IoT-edge or WSN in setting up a smart-campus or smart-city, need to effectively cover a very wide area, e.g., of the order of several square kilometres, under a limited budget. In order to meet the constraint, deployment of the devices often becomes highly non-uniform. Specifically, places of higher importance in a campus area such as buildings, auditoriums, etc., are usually covered densely with more devices whereas the connecting links, e.g., roads or streets are covered with much lesser density. Figure 1 shows a sample scenario of deployment of IoT-edge devices over the institute campus of IIT Bhubaneswar.

Fig. 1. State diagram of the proposed protocol FlexiCast.

In a decentralized system, communication protocols provide support as a backbone for carrying out all sorts of coordination and cooperation activities among the devices. However, the design and implementation of the existing protocols are evaluated in simulation platforms mostly over standard theoretical models of networks, e.g., 2D/3D grid or random structures with different degree distributions, etc. Publicly available WSN/IoT testbeds are usually considered to be more practical and meaningful for such evaluations. But they are mostly installed indoor and hence may not offer a high degree of skew. Communication protocols, thus, never get evaluated under highly non-uniform network structures that are commonly found in real-life systems.

Moreover, the design of the existing communication protocols, usually does not consider such diversity in the connectivity structure of the underlying network. For example, the well-known flooding protocol FTSP [1] would naturally incur high convergence time and reliability issues under such real-life networks having narrow paths and high diameter. Similarly, data collection

protocol CTP [2] would exhibit quite a high average latency for end-to-end data delivery, along with reliability issues in such networks. In recent works synchronous transmission (referred to as ST) based data sharing strategies [3] have gained quite good popularity because of their ability to achieve high reliability under very low-latency and low energy consumption. However, ST based protocols also may get affected considerably under such structural adversity. For example, the reliability of the protocol Glossy [4] would get affected if its duration and the parameter that controls the number of times a node transmits the packet (i.e., N-TX) are not decided appropriately based on the structure of the network. Similarly, the protocol LWB [5] also would get affected due to high diameter and a corresponding rise in the slot-time. Even the protocol Chaos [6] for all-to-all data sharing would get heavily affected in the regions of the network where density is high.

While large networks with large diameters are always detrimental to communication protocols, a suitable design considering the underlying network structure can help reduce these effects to a great extent. A design that includes flexibility and self-adjustment features, would bring an overall improvement in the performance of the protocols by ensuring that the target task is carried out in an optimal way under any network setting, not just in skewed networks. For example, a network for sensing a dangerous gas leak would be able to share the sensed data to all other nodes faster and more efficiently by organizing transmissions from each node at the right moment as well as for the right number of times depending on its position in the network. A swarm of drones could autonomously monitor their relative positions and status [7] by sharing the data with all the other drones in the herd, and the underlying protocol would self-adjust in real-time to new changes in positions to find a superior schedule for performing the data sharing task. However, most of the existing communication protocols lack these properties. In this work we propose a novel protocol *FlexiCast* for many-to-many data sharing which applies the concept of adaptability by building upon an existing inflexible protocol MiniCast [8].

In summary, the contributions from the work are as follows.

- We introduce the concept of a class of protocols that can self-adjust based on the specific structure of the underlying network in order to optimize its performance.
- We introduce a simple way for controlled run-time variation of physical layer parameters such as packet-size, length of packet-sequence, etc., which are generally required to be rigid for ST to work.
- We build a protocol *FlexiCast* that uses a time-varying TDMA schedule to carry out efficient

many-to-many data sharing in a wide variety of network structures. A full system is designed which starts with the protocol MiniCast and transits to FlexiCast in short time for an optimized performance under a given network setting.

- A comprehensive theoretical study of the performance difference between FlexiCast and MiniCast is provided.
- FlexiCast is implemented in Contiki OS for TelosB devices and is evaluated extensively in Cooja simulator as well as publicly available IoT/WSN testbeds.
- The performance of FlexiCast is compared with three state-of-the-art protocols MiniCast, Chaos and LWB.

The paper is organized as follows. Section II provides a brief background and the related works. Section III discusses the basic idea and design of the proposed protocol. Section IV describes the metrics used for measurement and performance comparison. Section V provides a theoretical study of the proposed design. Section VI provides an in-depth evaluation study of the proposed protocol in simulation as well as testbed settings. Finally, we conclude our work in Section VII.

II. BACKGROUND AND RELATED WORKS

ST based protocols in the domain of IoT/WSN have been gaining popularity because of their ability to achieve the goals with low latency, high reliability and high throughput. Most of the protocols under ST, are built on the design of Glossy [4]. Glossy achieves a very fast one-to-many data sharing and implicit time synchronisation across all the nodes in large networks. In summary, unlike traditional CSMA based strategies, Glossy completely avoids inter-packet collision by enabling multiple transmitters to transmit packets exactly at the same time. The simultaneous transmissions are scheduled in a way so that they result in constructive interference or capture effect for which the packets get successfully received. BlueFlood [9], SCIF [10], RedFixHop [11], LiM [12], Splash [13], Pando [14], [15], Ripple [16] etc., are all extensions over the same base protocol Glossy. However, a very less number of works are there which try to optimize the performance in run-time. One recent work [17] applies run-time learning and updates of the parameters of Glossy so that its performance can get better. Similarly, the work [18] facilitates dynamic updates of the MAC layer parameters as per requirements. However, these works do not target any network structure-specific optimization.

To facilitate run-time time optimization, a communication protocol, needs to be flexible enough. However, the protocols based on ST, inherently bear strong rigidity in important physical or data link layer parameters such as packet-length, schedule length etc. Moreover, network-specific optimization in a communication protocol needs a reflection of the network structure on some key attributes of the protocol. By the virtue of microsecond level perfect time-synchronization, ST based protocols make it possible to exploit even network-wide TDMA schedule to allow transmission of different data from all the nodes. The structure of a network can largely affect the performance of the protocol if such TDMA schedules are not appropriately formed.

Communication protocols, supporting many-to-many data sharing [3], serve a significant role in general in smart-systems. The framework of Glossy has been exploited in the design of many such protocols. Schedule based sequential operations are quite common in these protocols. However, there are variations in the way the schedules are applied. The works LWB [5], WTSP [19], Blink [20], Crystal [21] etc., apply TDMA schedule in global level for complete and independent repetition of Glossy floods. Hence, the scope of change in the behavior in the protocols through manipulation of the schedule is not possible. Protocols such as Chaos [6], A2 [22], Mixer [23], CodeCast [24], ByteCast [25] etc., use the schedule in a very granular way inside a packet and hence scope for adjustment of the schedules/packet size in skewed networks is there. The work MiniCast [8], composes a TDMA schedule with the packets transmitted from different nodes in a specific order. A proper balance between scheduling and granularity brings a very good scope in the protocol to get tuned.

The concept of run-time variation of TDMA schedules for network-specific optimisation is introduced in our previous work [26] in this direction. The work uses custom-made TDMA schedules and simulate the concept over standard multi-hop networks. In the current work, we develop a full system that can automatically build up the TDMA schedules from the network-structure related information collected in the nodes and also dynamically update those schedules as per the changes detected in the network-structure. In addition, we provide a theoretical derivation of key metrics and improvements and extensively test the underlying concept and the designed protocol in a wide variety of real as well as simulated networks. In the following, we first provide a brief architecture of MiniCast followed by the design of FlexiCast.

Overview of MiniCast: MiniCast follows a radio-event driven design like the protocol Glossy. The many-to-many dissemination process starts with the initiator node broadcasting the first set

of packets which triggers the first-hop nodes to transmit their data. Transmissions from the first-hop nodes trigger the transmission from the second hop nodes. The process continues until all the nodes complete a predefined number (N -TX) of transmissions. In order to facilitate many-to-many dissemination in a compact form, all transmissions in MiniCast happen in a chain of packets based on a globally fixed TDMA schedule. In a system comprising of N nodes, the schedule provides a unique position to each node to transmit its data packet. A *slot* in MiniCast accommodates the transmission (or reception) time for all the packets as per the schedule. Time for a full physical layer packet transmission is referred to as a *sub-slot*. In order to achieve precise timing at each sub-slot during transmission of a *chain* of packets, MiniCast exploits SFD interrupts during transmissions in addition to the SFD interrupts during receptions. Hence, unlike Glossy, in MiniCast a node does not need to interact with other nodes to achieve transmission-time synchronization. This is referred to as *self-synchronisation*. Each node in MiniCast starts with its own data to be shared. Gradually as the dissemination proceeds, the nodes acquire data from the other nodes. However, for self-synchronisation to work, a node has to transmit in every sub-slot even if the node doesn't have the necessary data yet. These dummy transmissions are done with the lowest possible power level helping other valid transmissions to get received appropriately in the receivers by the virtue of capture-effect.

ChainTx: Transmission of a chain of packets exploiting self-synchronization is a very fundamental unit of MiniCast. We extract out this basic unit from MiniCast and call it *ChainTx*. In principle, it is the same as Glossy and can be used for one-to-all dissemination. But instead of disseminating one packet from the initiator to the network like in Glossy, ChainTx transmits a chain of packets back-to-back.

III. DESIGN AND IMPLEMENTATION

The dissemination process in MiniCast starts from a single initiator node. In a usual network setting, most of the nodes are discovered within few hops. However, as discussed in Section I, in a cost-optimized IoT-edge deployment covering a large area, the connectivity structure of the underlying network may have dense components connected by narrow linking structures. Under such settings, the number of newly discovered nodes, w.r.t the initiator, grows very slowly while moving along the narrow strings and increases suddenly as a cluster arrives. Under such circumstances, the use of a globally fixed and large schedule at every execution step, as in MiniCast, would incur a huge wastage of time and energy as many sub-slots remain empty and

are simply transmitted with random data for a substantial amount of time. Run-time variation of the schedule is a necessary step to mitigate this problem. However, there are certain challenges in implementing the same under ST based protocols which we first address in the following.

Fig. 2. ChainTx with fixed[a] and variable[b] chain lengths

A. Run-Time Variation of Schedule-Size

ST based strategies [3] have been proved to be many times faster and energy-efficient than asynchronous transmission based strategies. However, in order to maintain precise time synchronization among the nodes and ensure constructive interference/capture effect during execution, various restrictions are strictly followed. For example, multiple nodes are allowed to transmit packets together but exactly at the same time (with microsecond level precision) and should have exactly the same size and content. The work Chaos [6], depends more on capture effect and hence it relaxes the restriction on same contents, but it has the same restrictions on packet size.

In this work, we exploit a special parameter called *relay-count* (aka slot-number or RC) to bring flexibility in the protocols in a very simple way. The protocol Glossy [4] introduces this parameter to keep track of the number of slots passed during its execution. Relay-count is initially set to zero in every node. Before transmission, the local value of the relay-count is incremented and placed in the packet. In MiniCast, the same strategy is followed. The value of the relay-count stays the same for an entire slot. Each sub-slot is identified by another *packet counter*. One very important property of relay-count in any ST based protocol is that at any point in time all the packets being transmitted throughout the network bear the same value of the relay-count

irrespective of the location of the transmitters. This allows relay-count to be used as a global reference time. We exploit this feature to realize the time-varying schedule in MiniCast and ChainTx.

Figure 2 graphically shows the feasibility of the strategy through execution of ChainTx with such variation in the chain-length in a network consisting of three layers w.r.t initiator. We assume a common predefined map that defines a different chain-length for a different relay-count. It can be seen from the figure that despite this variation, at different time point the length of schedule remains constant throughout the network which ensures successful ST. Following the same strategy, packet size can be also varied over sub-slots. We modify Glossy and ChainTx to accommodate this flexibility in the chain length and packet size. The modified versions are evaluated rigorously and no change in their reliability is observed (detailed in Section VI-A).

State diagram: The state diagram of FlexiCast is given in Figure 3. It bears a transmission loop and a reception loop to carry out the series of transmission and reception of packets. Before the transmission loop, the relay-count is incremented and the schedule for the slot is picked from the predefined map. Transmitters check the availability of the data before every packet transmission and receivers stores the data after every reception in a local *data-store*.

In FlexiCast, we introduce a new constant $K\text{-TX}$. A node terminates the protocol after performing a total of $N\text{-TX}$ transmissions or $K\text{-TX}$ transmissions after receiving the complete data, whichever happens first.

B. Formation of Time-Varying Schedule

Since the dissemination process starts from an initiator node, the number of hops of nodes discovered during initial slots is quite limited. Hence, in general, at slot number i there is no point in including the nodes beyond hop i in the schedule. For example, during the execution of MiniCast in the network shown in Figure 5, in slot-number 2, the schedule does not need to consider nodes in layer 2 or higher since there is no way those nodes can contribute anything at that point. Thus, the TDMA schedule should be appropriately formed to address these issues. However such issues are directly linked with the structure of the underlying network.

To understand how MiniCast itself can be useful for obtaining the necessary information, we run MiniCast for 100 iterations in a small 15-node sub-network of testbed DCube [27] as shown in Figure 4. Analysing the logs we find that, despite the error-prone wireless transmissions, in a large fraction of the iterations, the data from a node at i th hop is received at the initiator only

Fig. 3. State diagram of the proposed protocol FlexiCast.

starting from the i th reception. In some iterations, it got deferred to next $(i + 1)$ th reception slot. We use this chance of reception of data at the initiator (referred to as *reception-probability* and abbreviated as *RP*) as a means of categorizing the nodes into different hops ($h_1, h_2, h_3, \dots$) as shown in Figure 4 with their RPs. A similar experiment was done for a simulated skewed network shown in Figure 5. **Using the RP values, we can say that a node is in i^{th} hop with a probability equal to its i^{th} RP.**

Fig. 4. RPs for predicted hops in a uniform 15-node network selected from DCube. Approximate layer distances mentioned with dotted circles. Every node within or on the boundary of first dotted circle is considered to be in first hop.

Fig. 5. Predicted hops and its RP for nodes of a skewed network created in simulation.

However, due to the error-prone nature of wireless transmission, sharp transition in the RPs may not be always visible. As shown in Figure 4 some nodes have intermediate RPs such as 0.57, 0.43 etc. Deciding the right hop number for these nodes is crucial. If such a node is pushed back, other nodes that are depending on it may get victimized. Whereas if it's brought forward, it may unnecessarily eat up time due to empty sub-slots. We go for the latter option as its comparatively a safer solution and hence set a threshold Th_p with a moderate value of 0.4 on the RPs to decide the layer of the node. Once the layers are decided, for the i^{th} transmission all nodes till i^{th} layer will get a slot (tagged with node ids) in the TDMA schedule in any order.

C. Full System

Figure 6 provides a schematic of the flow of the execution of the full system. The process begins with the initiator executing a certain number of iterations of MiniCast with the full TDMA schedule. The RPs are calculated and the time-varying schedule is derived accordingly. The RPs are preserved for future references. They are referred to as *Reference Reception-Probabilities* (RRP). Subsequently, the initiator disseminates the schedule to all the nodes in the network using ChainTx. FlexiCast starts from the very next iteration after reception of the time-varying schedule at all nodes. However, in order to track for possible changes in the network structure, the RPs are calculated and compared with RRP in the initiator node in FlexiCast. A set of possible cases are listed below.

- 1) *A node gets shifted to a new location in the same hop*: It cannot be detected. However, it won't affect the process in any way.
- 2) *A node gets pushed further away from the initiator*: The initiator can detect it even with a fair idea about how many hops it got shifted. Initiator recomputes the new schedule and disseminates again.
- 3) *A node comes closer to the initiator*: Initiator will observe an increased RP for this node. It would recompute the schedule.
- 4) *Node failure*: This can be well detected by the initiator. This can happen if there is a certain change in the network, e.g., reorientation etc.
- 5) *Node addition*: Its not detected automatically, but usually in such cases the initiator is supplied with the id of the new node(s).

Under cases 2 and 4, initiator recomputes the time-varying schedule and disseminates the new schedule to the nodes. Whereas, under cases 3 and 5, the schedule needs to re-formed from scratch through explicit execution of MiniCast with full schedule for certain number of iterations. These cases are shown in Figure 6 as *small change* and *significant change*.

IV. METRICS

The performance of FlexiCast is compared with state-of-the-art many-to-many data sharing protocols Chaos and LWB in addition to its base protocol MiniCast. Experiments are compared based on the following metrics.

Fig. 6. The overall execution flow of the proposed system.

Information Coverage: We mostly consider all-to-all data sharing where each node has some amount of data to share with all other nodes in the network. The percentage of data collected at every node after each reception-slot is measured and is referred to as information coverage.

Latency: Latency is defined as the time taken for a certain percentage of information coverage in each node.

Radio-on time: It is the total amount of time the radio in a node is required to remain ‘ON’ to complete a single iteration of the process. It represents energy consumption.

There are few other metrics, which are specific to the context and are described in the places they are used in the evaluation section. Every experiment with FlexiCast and all other protocols is composed of 1000 iterations where one iteration refers to a complete run of the protocol over the entire network. All the results presented are calculated by averaging the values of the metrics separately over all the participating nodes, and then over all iterations. The error bars in the results reflect the standard deviation values across iterations.

Network structure: A primary target of this work is to achieve efficient many-to-many data sharing under skewed networks, which are modelled as a combination of strings and clusters as shown in Figure 1. However, in WSN/IoT testbeds, it is hard to get such structures. Hence, we experiment on appropriately formed sub-networks of these testbeds. In addition, in order to carry

out an in-depth study, we simulate many different network structures in the Cooja simulator.

V. THEORETICAL STUDY

In this section, we build up a theoretical understanding of the proposed strategy. In both MiniCast and FlexiCast, the transmissions are carried out in chains of packets. Hence, we first calculate the transmission time of a packet and then the same for the i_{th} chain in both of the protocols. These are then used for theoretically comparing their latency and the radio-on time. In the following discussion we abbreviate *transmission-time* as *tx-time*.

Tx-time of a packet: In MiniCast, in a network with n nodes, every transmission after initiation consists of n body packets, one header and one trailer packet. The packets in both FlexiCast and MiniCast are composed of a length field of 1 Byte, a packet type field of 1 Byte, schedule information of 2 Bytes, data of d Bytes, and the standard trailer field of 2 Bytes, i.e., total $(6+d)$ Bytes [8]. In addition, a packet transmission would also involve 4 Bytes of calibration time, 4 Bytes of preamble (zero symbols), 1 Byte of SFD leading to a total $T_p = (15 + d) \times T_b$ as the effective tx-time for a single packet, where T_b is the tx-time of a single Byte.

Tx-time of a chain: While the header and trailer packets have no data, i.e $d = 0$, and hence have a constant size, for simplicity in this work we assume that they also have the same size as the body packets. Thus, the tx-time of a packet in MiniCast can be computed as,

$$T_{MiniCast} = T_p(n + 2). \quad (1)$$

In FlexiCast, the length of the schedule varies with relay-count. The schedule at relay-count i considers all nodes that are reached upto that time-point. Let $g(k)$ define the number of nodes present at a hop-distance k from the initiator, and h be the total number of hops present in the network. Thus, length for i_{th} chain (schedule length at relay-count i), $f(i)$, can be defined as the cumulative sum of number of nodes till hop i , i.e., $\sum_{k=0}^i g(k)$. Note that for relay-count $i \geq h$ would contain all nodes till last hop h , i.e., the entire network. Thus, the effective tx-time of a chain at relay-count i would be,

$$T_{i(FlexiCast)} = T_p(f(i) + 2) = T_p \sum_{k=0}^i g(\min(k, h)) + 2T_p. \quad (2)$$

For large networks, the constant time taken by header and trailer packets would be negligible. Equation 1 and Equation 2 can be simplified as follows.

$$T_{MiniCast} = T_p \times n, \quad (3)$$

and,

$$T_{i(FlexiCast)} = T_p \times \sum_{k=0}^i g(\min(k, h)). \quad (4)$$

Schedule size at relay-count i in MiniCast is an upper limit of the same in FlexiCast. In the formation of the schedules in FlexiCast, distribution of the nodes described by $g(k)$ plays an important role. However, during the initial runs $g(k)$ is not known. Hence, as shown in Section III, the reception statistics at the initiator in MiniCast are used to construct $g(k)$.

Latency: Network structure plays a very important role in the performance of the proposed protocol. Let us consider a string like network as shown in Figure 5. Every node in hop i is connected to every node in hop $i - 1$, ensuring that all data collected till hop $i - 1$ is forwarded to all of hop i . In such a network the first transmission from the nodes at the last hop (h) would complete the data collection at hop $h - 1$, and each transmission from here on would forward the data hop-by-hop. Transmission of the chains with relay-count $i \geq h$ would need to use the full schedule, hence, would be same as that of MiniCast, i.e., $T_{MiniCast}$. Thus, the latency for a node at hop i in FlexiCast can be calculated as,

$$L_{i(FlexiCast)} = \sum_{i=0}^h T_{i(FlexiCast)} + |h - 1 - i| \times T_{MiniCast}. \quad (5)$$

Assuming a large network, with large h , Equation 5 can be simplified as

$$L_{i(FlexiCast)} = \sum_{i=0}^h T_{i(FlexiCast)} + |h - i| \times T_{MiniCast}. \quad (6)$$

In MiniCast, the first transmission from the initiator has one body packet, while all others chains will have n body packets. Hence, the latency of a node at hop i in MiniCast can be calculated as,

$$L_{i(MiniCast)} = (T_p * 3) + (h + |h - 1 - i|) \times T_{MiniCast}. \quad (7)$$

Assuming a large network, with large h , it can be simplified as,

$$L_{i(MiniCast)} = (h + |h - i|)T_{MiniCast}. \quad (8)$$

The improvement in FlexiCast comes from the transmission of the first h chains having much smaller schedule sizes. For a simple uniform distribution with m nodes per hop, let us consider this part of Latency as L' . For FlexiCast, from Equation 4 we get,

$$L'_{FlexiCast} = \sum_{i=0}^h T_p \times \sum_{k=0}^i g(k) = \sum_{i=0}^h T_p \times m * i = T_p \times m \times \frac{h \times (h + 1)}{2} \quad (9)$$

The same for MiniCast from Equation 3 would be,

$$L'_{MiniCast} = T_{MiniCast} \times h = T_p \times n \times h = T_p \times m \times h \times h \quad (10)$$

$L'_{MiniCast}$ is thus roughly double of $L'_{FlexiCast}$. For skewed distribution of the nodes at different hops, e.g., $g(k) = k^2$, or having a cluster at the end as shown in Figure 8(a), there will be further improvements in the latency in FlexiCast.

Radio-On Time: A node in FlexiCast is required to complete either K-TX (used as ktx in the equations) transmissions after receiving all data or N-TX (used as ntx in the equations) transmissions in total, whichever is less. Both N-TX and K-TX are programmable to assist faster completion of the protocol. Since in FlexiCast, the hops are properly defined, we found that a few transmissions from the nodes after their completion is sufficient for forwarding all the data. So, radio-on time of a node at hop i would be,

$$R_{i(FlexiCast)} = L_{i(FlexiCast)} + (2ktx - 1) \times T_{MiniCast}. \quad (11)$$

The term $(2ktx - 1)$ reflects the fact that ktx transmissions would result in ktx-1 receptions also. In case of MiniCast, the nodes need to perform N-TX transmissions in total, so it does not depend on the latency. Thus, radio-on time of a node at hop i can be calculated as,

$$R_{i(MiniCast)} = T_{MiniCast} \times (i - 1) + (2ntx - 1) \times T_{MiniCast}. \quad (12)$$

The term $(2ntx - 1)$ reflects the fact that for ntx transmissions there will be ntx-1 receptions after the first reception for a node at hop i .

In MiniCast, radio-on time doesn't depend on body packet failures. However, in FlexiCast, the completion depends on achieving 100% information coverage in the nodes, requiring all body packets to be successfully received at least once. The nodes will seldom make a total of N-TX transmissions before turning their radio off and in those cases the radio-on time of FlexiCast will be equal to that of MiniCast. Section VI provides a comparison of the performance of the two protocols in WSN/IoT testbeds and tallies with these theoretical observations.

VI. EVALUATION

In this section, we first show the effectiveness of the relay-count based solution for implanting flexibility in ST based protocols as described in Section III. Next, a detailed simulation based study of FlexiCast is provided followed by an evaluation study over publicly available testbeds DCube and Indriya. In all our experiments the value of N-TX is set to 10, K-TX in FlexiCast is set to 2 (see Section IIIA) and Th_p to 0.4 (see Section IIIB). The slot-time in LWB is set to 15 ms unless explicitly specified. To make the comparisons among the protocols easy, we share 1-byte data from each node unless specified. Increase in data size brings about more improvement in FlexiCast, which we show later in the same section. The protocol can be easily scaled to accommodate sharing of larger data by splitting data among multiple rounds of the protocol, each round taking a few milliseconds.

A. Flexibility in Synchronous-Transmission

Glossy and ChainTx originally use fixed packet size and fixed chain length. In our experiments, we refer to them as *Glossy with Fixed Packet Size (GFPS)* and ChainTx with Fixed Chain Length (CFCL). We modify the base protocols and introduce the following two versions: Glossy with Varying Packet Size (GVPS), and ChainTx with Varying Chain Length (CVCL). In the following, we first evaluate the feasibility for relay-count based run-time variation of physical layer parameters such as packet size, packet content, length of chain etc., independently through explicit experiments over all these four different protocols. The metric *Reliability-rx*, described below, is used as the primary metric for comparison here.

Reliability-rx: In Glossy [4], a node gets N-TX number of chances to receive a packet throughout the execution of a single round of dissemination process. In Glossy, an iteration is considered to be successful (i.e., reliability 100%) if a node receives at least once out of these N-TX chances. For the current set of experiments, however, we consider each reception to be important and observe how controlled variation in physical layer parameters affects the same. *Reliability-rx* is calculated as a ratio of *the number of the packets a node successfully receives to the total number of chances it gets to receive packets*.

All the four strategies are tested in the testbeds Indriya [28] and DCube [27]. Figure 2 presents the average reliability-rx from the DCube. The line graph in 7(a) shows the reliability-rx in GFPS. It can be noticed that reliability-rx drops with the increase in the packet size. It happens due to higher accumulation of clock-drifts with higher packet size resulting in inaccurate time

Fig. 7. (a) Reliability-rx in GFPS, GVPS and CFCL (b) Reliability-rx in CVCL as described in Section VI-A

synchronization. However, original reliability still remains 100% regardless of the packet size. In the case of GVPS, we generate a random sequence of packet sizes within a (min, max) range. Figure 7(a) shows the average reliability-rx in this experiment w.r.t the mean of this random sequence as bars. Its visible from the figure that the bars match closely with the line graph representing reliability in GFPS indicating no considerable drop in GVPS. In CFCL, we fix the chain-length at 20 and vary the packet size. We observe in the dashed line graph of Figure 7(a) that the reliability-rx values do not degrade much with back-to-back chain based transmissions of packets. In CVCL, we vary the chain length in each slot while all packets bear a fixed size. We generate five random sequences of chain lengths in the same way as in GVPS experiments. Reliability-rx is plotted w.r.t the average chain in Figure 7(b) for three different packet sizes. Comparing Figure 7(a) and Figure 7(b) it can be checked that the average reliability-rx for a certain packet size in CVCL matches almost perfectly with the same for the same packet size in CFCL. Reliability-rx in CVCL also matches with the cases of similar average packet size in GVPS and GFPS.

In a nutshell, all these results establish that the proposed strategy is quite feasible and effective under ST without incurring any degradation in reliability.

B. Simulation Based Study of FlexiCast

For simple single-hop and multi-hop networks, FlexiCast performs at least as good as MiniCast with some improvement in multi-hop networks depending on the distribution of nodes among

Fig. 8. Network structures modeled as combinations of strings and clusters for simulation. The dark node is the initiator.

Fig. 9. Radio-on time and latency of FlexiCast, MiniCast, Chaos and LWB in networks of type 8(a) with $n=30$ and $s=3, 6, 9$.

hops [26]. For modelling skewed networks, we use network structures as shown in Figure 8(a).

Networks composed of string and cluster, with varying string length: We fix a single-hop 30-node cluster and vary the string lengths to 3, 6 and 9 hops. The results are show in Figures 9(a,b), (c,d), (e,f) respectively. For the network with 9 hops, we set LWB slot time to 20 ms to accommodate for larger diameter. In these configurations, increase in the diameter causes the

most adverse effects. LWB's performance highly depends on slot-time which in turn depends on the network diameter. A sudden increase in latency and radio-on time is, hence, visible for the network having a 9-hop string. The network becomes more sparse due to the increase in the string length, which benefits Chaos. But increase in the number of nodes leads to an increase in packet size also which degrades its reliability. Hence, on balance, Chaos degrades slightly. MiniCast degrades with the string length since the protocol takes more full-size chain receptions to complete. FlexiCast, through the use of its optimised schedules, saves a lot of time and energy. In particular, it takes about 30%, 43% and 47% lesser latency as well as 26%, 49% and 58% lesser radio-on time compared to MiniCast to achieve 100% all-to-all dissemination with string lengths 3, 6, and 9 hops, respectively.

The network contains a total of 43 nodes spread over 7 hops and 12 nodes form string of length 6 and width 2. Using equations 5 and 2 in Section V the latency of a node at hop 6, can be derived as $\sum_{i=0}^h T_{i(FlexiCast)} = T_p \times (1+3+5+7+9+11+13+43) + 2 \times T_p \times 8 = 1608 \times T_b$. Each transmission from this point would take $T_p \times (43 + 2) = 720 \times T_b$. With each of these full transmissions, 100% information coverage will be achieved hop by hop. Thus, the latency value averaged over all the 43 nodes would be $L_{avg(FlexiCast)} = 1608 \times T_b + T_b \times \frac{720 \times (30 + 2 \times (1+2+3+4+5)+1)}{43} = 2629.4 \times T_b$. In case of MiniCast, using equations 7 and 1 the latency for nodes at 6th hop would be $\sum_{i=0}^h T_{MiniCast} = T_p \times (1+43 \times 7) + T_p \times 16 = 5088_b$. The average latency would be $L_{avg(MiniCast)} = 5088_b + T_b \times \frac{720 \times (30 + 2 \times (1+2+3+4+5)+1)}{43} = 6109.4 \times T_b$. Thus, theoretically, FlexiCast achieves 57% lesser latency with 100% reliable transmissions. However, the improvement we get in the experiment is 43% which is while similar, is little lesser than the theoretical finding. This happens because every failure in the protocol adds a transmission of the full chain, which makes FlexiCast slightly more sensitive, and hence improvement goes little down than what is expected.

Using equation 11 the average radio-on time for FlexiCast can be derived as $R_{avg(FlexiCast)} = 4789.4 \times T_b$. While for MiniCast, the average radio-on time from equations 12 and 1 would be $R_{avg(MiniCast)} = \left(\frac{46 \times 2 + (46 + 720) \times (2 \times (1+2+3+4+5) + 6 \times 30)}{43} \right) + (2ntx - 1) \times 720_b = 17420.3 \times T_b$. Thus, with 100% reliable transmissions, FlexiCast achieves 72% lesser radio-on time compared to MiniCast. However, in the experiment, we get an improvement of 49%. Radio-on time of MiniCast is depends only on header and trailer packet failures and isn't affected by the failures of the transmission of the body packets. However, radio-on time in FlexiCast is dependant on latency, which is dependant on the failures. Any body packet failure will delay the latency till

Fig. 10. Radio-on time and latency of FlexiCast, Minicast, Chaos and LWB on networks of type 8(a) with a s=6 and n=10,30,50.

the reception of the next chain. On top of this, a trailer packet failure would mark the reception a even if data in body packets were received. Therefore, in a realistic situation, if we assume on average 2 more chain receptions (and transmissions in between) for successfully receiving all body packets, we see the radio-on time is $4789.4 + 4 \times 720 \times T_b = 7669.4 \times T_b$ leading to a 55% improvement which is comparable to the experimental value we observe.

FlexiCast also defeats Chaos by completing all-to-all dissemination 82%, 74% and 64% faster while consuming 70%, 60% and 50% lesser radio-on time, respectively.

Networks composed of string and cluster, with varying cluster size: Fixing the string length at 6 hops, we vary cluster sizes to 10, 30 and 50 nodes. Results are shown in Figures 9(a,b), (c,d), and (e,f), respectively.¹

As the size of the cluster grows, the number of nodes increases while the diameter is kept constant. In LWB, slots take the same time for all cluster sizes but more slots are needed for more nodes. Hence, it increases linearly with respect to the no of nodes. In Chaos, proliferation in the dense cluster leads to more failure of natural capture effect. Under the larger size of the cluster, thus, the performance of Chaos degrades heavily in cluster size 50. In the case of MiniCast, a larger global schedule size to accommodate all nodes is used for every transmission. In case of FlexiCast, schedule size increases only after the cluster is discovered and the initial transmissions

¹For the sake of continuity and clarity, Figures 9(c,d) are repeated in Figures 10(c,d)

were done with much smaller schedules. In particular, FlexiCast takes 34%, 43% and 49% lesser latency, and 40%, 49% and 53% lesser radio-on time, compared to MiniCast, in the configurations having cluster sizes 10, 30, and 50, respectively. To show that these improvements would only increase with the scale of the network, we perform experiments with huge clusters of sizes 100, 150, 200 keeping string length at 4. We can see that MiniCast takes 157%, 170% and 177% more in latency as well as 176%, 183% and 187% more in radio-on time compared to MiniCast to achieve 100% all-to-all dissemination, respectively.

The network shown in Figure 8(a) is spread over 5 hops. It contains a cluster of 150 nodes and a string of length 4 having width 2. Thus, total number of nodes is 159 ($= 1 + 4 \times 2 + 150$). Using equations 5 and 2 for latency of a node at hop i , $L_{i(FlexiCast)}$ as derived in Section VIB, the latency for a node at 4^{th} hop would be $\sum_{i=0}^h T_{i(FlexiCast)} = T_p \times (1 + 3 + 5 + 7 + 9 + 150) + 2 \times T_p \times 6 = 2992 \times T_b$. Each transmission from this point would take $T_p \times 159 = 2574_b$ making the average latency, $L_{avg(FlexiCast)} = 2992 \times T_b + T_b \times \frac{2574 \times (150 + 2 \times (1 + 2 + 3) + 1)}{159} = 5630.75 \times T_b$. In case of MiniCast, the latency for a node at 4^{th} hop would be $\sum_{i=0}^h T_{MiniCast} = T_p \times (1 + 159 \times 5) + 2 \times T_p \times 6 = 12928 \times T_b$. Thus, the average latency of MiniCast $L_{avg(MiniCast)} = 12928 \times T_b + T_b \times \frac{2574 \times 163}{159} = 15566.75 \times T_b$. Hence, theoretically, latency in MiniCast is 176% more compared to FlexiCast, which matches quite well with the observed improvement.

Using Equation 11 as derived in Section VIB, the average radio-on time for FlexiCast can be computed as $R_{avg(flexicast)} = 13352.75 \times T_b$. By Equation 12, the average radio-on for MiniCast would be $R_{avg(minicast)} = \left(\frac{(46 + 2574) \times (2 \times (1 + 2 + 3) + 4 \times 150)}{159} + (2 \times ntx - 1) \times 2574 \right) \times T_b = 58990.7 \times T_b$. Thus, theoretically, FlexiCast takes 4.42 times less radio-on time compared to MiniCast with 100% reliable transmissions. As discussed in our previous example, if we consider two more chain transmissions for successful coverage in realistic scenario, we get $R_{avg(FlexiCast)} = 2574 \times 4 + 13352.75 = 23648.75 \times T_b$. In this case, MiniCast takes 150% more radio-on time. In our experiment it takes 183% more time, which is similar to the theoretical findings.

Varying data length: To further show the effect of increasing the length of data shared from each node, we use the same network structure with a string length of 3 and vary cluster sizes between 10, 20, 40 and 80. We also vary data lengths from 1 byte to 16 bytes and calculate the Latency and Radio-on time for 100% information coverage. The complete set of results are shown in Figure 11. From the inset, we can see that the improvement in FlexiCast increases

Fig. 11. Latency and Radio on time when varying data length in networks of type 9(a) with $s=4$ and $n=10,20,40,80$. The inset shows fraction of time taken by FlexiCast w.r.t MiniCast.

with an increase in data-length. Even in a large network of 87 nodes, FlexiCast takes only 30% of latency time and 50% of radio-on time that MiniCast takes for the same many-to-many data sharing.

Networks composed of more than one cluster and string: Real life IoT/WSN deployments might have multiple clusters with different sizes, multiple strings, junctions, etc. Due to space limitations, here we discuss two such special structures as depicted in Figure 8(c) referred to as SP1, and 8(b) referred to as SP2. Results for both are shown in Figure 12(a) and Figure 12(b), respectively. We show latency to achieve different percentages of information coverage and inset shows radio-on time for 100% information coverage. In both SP1 and SP2, FlexiCast shows consistently better performance. Specifically, it shows a latency and radio-on time improvement of 30% and 30% in SP1 and around 34% and 23% in SP2 on average compared to MiniCast.

We also experiment with many other parameters such as cluster positions, strings widths, overall topology, etc. However, it is beyond the scope of the current paper to discuss all these issues.

C. Evaluation of FlexiCast in testbed

We select specific nodes within the testbed networks and try to define sub-networks with non-uniform/skewed structure. Two different sub-networks of DCube were selected for evaluation.

Fig. 12. Latency with respect to information coverage. Inset shows the radio-on time for achieving 100% information coverage on simulated networks 8(b) and 8(c).

Fig. 13. Latency and radio-on time (inset) in two sub-networks of testbed DCube. Sub-network in part (a) is a multi-hop cluster following a standard network model. FlexiCast performs as good as MiniCast. Sub-network in part (b) is of type 8(b) where FlexiCast shows significant improvement over MiniCast and all other protocols.

The first sub-network is a usual multi-hop cluster where FlexiCast works similar to MiniCast. The second one is a carefully selected 22-node sub-network of type 8(a) with a 6-hop string consisting of one node per-hop and a three-hop cluster of 16-nodes. Figure 13(a,b) show the latency for different percentages of information coverage and radio-on times for 100% information coverage in FlexiCast. Huge improvements in FlexiCast can be seen in this sub-network owing to its time varying schedule which changes as per the underlying network. Specifically, FlexiCast has latency

Fig. 14. Stabilization of the calculation of hop positions of some of the nodes under two different sub-networks of Indriya.

and radio-on time improvements of 37% and 45%, respectively, over MiniCast, 66% and 60%, respectively, over Chaos, 72% and 62%, respectively, over LWB.

In Indriya, we select a 16-node sub-network, having structure similar to 8(b) without the cluster in the junction point. Even under this small setting, FlexiCast exhibits an improvements of 23.8% in latency and 45.3% in radio-on time, compared to MiniCast. The formation of time-varying schedule is also evaluated in the testbed Indriya with the same sub-network. The time it takes for stabilization of the RPs derived from MiniCast executions depict how fast the data sharing process can switch from MiniCast to FlexiCast (see Figure 6). Figure 14(a) and 14(b) show the predicted hops for some nodes with multiple iterations of MiniCast run with two different initiators. The graph mentions the exact node id used in Indriya. It can be seen that for real networks, stabilization can be achieved within 10 to 15 iterations of MiniCast.

VII. CONCLUSION

Real-life deployments of sensor networks or IoT systems often show a non-uniform spatial distribution of the nodes. In this work, we show that performance of the communication protocols can be substantially improved if such structural property is appropriately used to tune the design of the protocols. We also show a generic way to make the ST based protocols flexible and suitable for run-time adjustments. We develop a complete system to achieve run-time optimization in an existing many-to-many communication protocol and demonstrate a huge improvement over the base protocol, especially under the aforementioned skewed network structures.

REFERENCES

- [1] M. Maróti, B. Kusy, G. Simon, and A. Ledeczi, “The flooding time synchronization protocol,” 01 2004, pp. 39–49.
- [2] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, “Collection tree protocol,” in *Proceedings of SenSys*. ACM, 2009, pp. 1–14.
- [3] M. Zimmerling, L. Mottola, and S. Santini, “Synchronous transmissions in low-power wireless: A survey of communication protocols and network services,” *arXiv preprint*, 2020.
- [4] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, “Efficient network flooding and time synchronization with glossy,” in *Proceedings of IPSN*.
- [5] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, “Low-power wireless bus,” in *Proceedings of SenSys*, 2012, pp. 1–14.
- [6] O. Landsiedel, F. Ferrari, and M. Zimmerling, “Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale,” in *Proceedings of SenSys*. ACM, 2013, pp. 1:1–1:14.
- [7] S. Bharadwaj, K. Gonabattula, S. Saha, C. Sarkar, and R. Raja, “Concurrent transmission for multi-robot coordination,” 2021.
- [8] S. Saha, O. Landsiedel, and M. C. Chan, “Efficient many-to-many data sharing using synchronous transmission and tdma,” in *13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*.
- [9] B. A. Nahas, S. Duquenooy, and O. Landsiedel, “Concurrent transmissions for multi-hop bluetooth 5,” in *Proceedings of EWSN*, 2019.
- [10] Y. Wang, Y. He, X. Mao, Y. Liu, and X. Li, “Exploiting constructive interference for scalable flooding in wireless networks,” *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 1880–1889, 2013.
- [11] “Redfixhop: Efficient ultra-low-latency network flooding,” in *Proceedings of the IEEE Int. Conference on Sensing, Communication, and Networking (SECON)*, 2016.
- [12] P. Zhang, A. Y. Gao, and O. Theel, “Less is more: Learning more with concurrent transmissions for energy-efficient flooding,” in *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2017.
- [13] M. Doddavenkatappa, M. C. Chan, and B. Leong, “Splash: Fast data dissemination with constructive interference in wireless sensor networks,” in *Proceedings of USENIX NSDI*, 2013.
- [14] W. Du, J. C. Liando, H. Zhang, and M. Li, “When pipelines meet fountain: Fast data dissemination in wireless sensor networks,” in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, 2015.
- [15] W. Du, J. Liando, and H. Zhang, “Pando: Fountain-enabled fast data dissemination with constructive interference,” *IEEE/ACM Transactions on Networking*, 2016.
- [16] D. Yuan and M. Hollick, “Ripple: High-throughput, reliable and energy-efficient network flooding in wireless sensor networks,” in *Proceedings of IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2015.
- [17] V. Poirot and O. Landsiedel, “Poster: Learning to shine – optimizing glossy at runtime with reinforcement learning,” in *Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks*, ser. EWSN '19. USA: Junction Publishing, 2019, p. 226–227.
- [18] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele, “ptunes: Runtime parameter adaptation for low-power mac protocols,” in *2012 ACM/IEEE 11th International Conference on Information Processing in Sensor Networks (IPSN)*, 2012, pp. 173–184.

- [19] M. Suzuki, C.-H. Liao, S. Ohara, K. Jinno, and H. Morikawa, "Wireless-transparent sensing," in *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, 2017, p. 66–77.
- [20] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele, "Adaptive real-time communication for wireless cyber-physical systems," *ACM Trans. Cyber-Phys. Syst.*, 2017.
- [21] T. Istomin, A. L. Murphy, G. P. Picco, and U. Raza, "Data prediction + synchronous transmissions = ultra-low power wireless sensor networks," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, 2016, p. 83–95.
- [22] B. Al Nahas, S. Duquennoy, and O. Landsiedel, "Network-wide consensus utilizing the capture effect in low-power wireless networks," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, 2017.
- [23] C. Herrmann, F. Mager, and M. Zimmerling, "Mixer: Efficient many-to-all broadcast in dynamic wireless mesh networks," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, 2018, p. 145–158.
- [24] M. Mohammad and M. C. Chan, "Codecast: Supporting data driven in-network processing for low-power wireless sensor networks," in *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2018, p. 72–83.
- [25] S. Saha and M. C. Chan, "Design and application of a many-to-one communication protocol," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.
- [26] M. Tummala and S. Saha, "Concurrent transmission based data sharing with run-time variation of tdma schedule," in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, 2020, pp. 461–464.
- [27] M. Schuß, C. A. Boano, M. Weber, and K. Römer, "A competition to push the dependability of low-power wireless protocols to the edge," in *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*. Junction Publishing, 2017.
- [28] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, "Indriya: A low-cost, 3d wireless sensor network testbed," in *Testbeds and Research Infrastructure Development of Networks and Communities*, 2012.