ISSN- 2394-5125 VOL 7, ISSUE 03, 2020

AN EFFICIENT BLUETOOTH SCATTERNET COMMUNICATION ALGORITHIMS FOR DYNAMIC ENVIRONMENTS

1.Ravi Kishore Veluri, 2.Dr.Neeraj Sharma 3.Dr.V.V.Krishna 4.Dr.S.Rama Sree, 5.Dr.Deepak Nedunuri
[1]Research Scholar, Sri Satya Sai University of Technology & Medical Sciences,Sehore
[2]Associate Professor, SSSUTMS,Sehore
[3]Professor in IT, Vasavi College of Engineering, Hyderabad.
[4]Professor in CSE ,Aditya Engineering College(A), Surampalem.
Email ID: ravikishorev1985@gmail.com

ABSTRACT
New and promising technologies, such as Bluetooth-based networking, are emerging and taking small-area networking to a higher and better level. With the support of the Bluetooth specification, piconet formation is encouraged. However, the use of the Scatternet remains available. The biggest problem with piconet formation is that they cannot be interconnected. This research presents a new, more efficient algorithm for Bluetooth mesh network formation. In a dynamic environment, where nodes show up and leave at random, our protocol will provide high performance. The incremental building of the TSF partitions occurs as soon as the topology and healing partitions are introduced. in the context of networking, we built a Bluetooth simulator that incorporates nearly all of the details of the stack of Bluetooth protocols.
Keywords: Bluetooth, Dynamic, Environment, Protocol.

I. OVERVIEW

Mobile devices have become an essential part of daily life for people who have benefitted from recent advances in wireless technology and mobile applications. Mobile libraries and navigation have been found in many smartphone apps. "More advanced wireless ad-hoc and sensor networks support mobility in the mobile application arena, because wireless PANs and sensor networks are possible on mobile devices that have no base stations or wired networks. Wireless sensor and ad hoc networks, such as healthcare, fitness, and gaming, are growing in use. Since short-range data transmission technologies such as Bluetooth enable easy data exchange between devices that are stationary or mobile, Bluetooth is a wireless protocol that utilizes short-range communication technologies. Due to the outstanding wireless properties, such as low cost, low power consumption, and good anti-jamming performance, Bluetooth has become an attractive communication protocol for wireless ad hoc and sensor networks. Each piconet consists of no more than eight devices because each device has three-bit addresses. Each piconet links a single member with another piconet that is also another member of the same piconet." This device connects two piconets together, and it transmits information between the members of both piconets. "All piconets in the network form a scatternet, allowing connected devices to interact with each other.

Finding the best network configuration, with minimal network reliability and connectivity issues, is a difficult task for a scatternet formation algorithm. When it comes to sensor networks using Bluetooth, the number of bridges and the connectivity of each bridge will affect how long the sensor network functions and how long it survives. Using more bridges has the net benefit of an increase in overall network efficiency, and decreasing the total data transfer path." The bridge node, however, is liable for forwarding packets between two or more piconets, which happens in a time-shared manner [1]. In other words, since the network requires more resources to maintain, the bridge nodes have a shorter lifespan.

II. FRAMEWORK OF SCATTERNET

Scatternet is a network of interconnected piconets. An interlinked network such as this will improve the tractability of networking, while at the same time, it will speed up the creation of new applications. As an illustration of this, imagine that you see an old man in your local public library. A figure can be visualized in different ways, depending on the lens, perspective, and interpretation. The piconets form a scatternet when connected together. A peer-to-peer link is established in Scatternet. It is established when a member of one piconet behaves as a slave in another piconet. Piconets acting as a bridge can be handled by a single device. A participant can be either a master or a slave. A device can serve as a slave in more than one piconet. It serves as a central device, but it only functions in one piconet [2]. The computer is present in two piconets, and so it functions as a broadcast transmitter. This intermediate device distributes data packets across the piconets based on time division multiplexing (TDM). A system's ability to contribute to scatternet communication depends on

ISSN- 2394-5125 VOL 7, ISSUE 03, 2020

having the ability to access multiple points. When Scatternet expands communication between more than eight devices, it creates a network with more devices. It efficiently utilizes bandwidth.

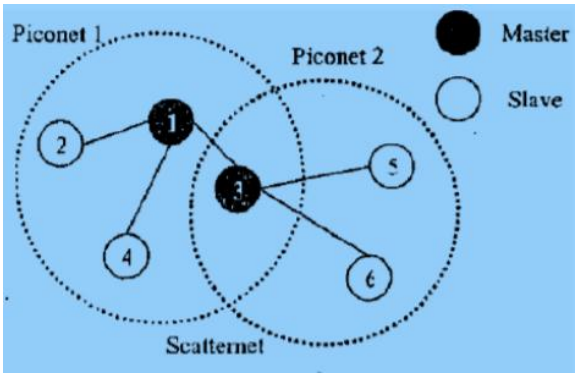


Figure 1: Scatternet formed by overlapping of piconets

- **Scatternet topology**

To date, there is little relevant research on how to build Bluetooth scatternets. "The scatternet creation schemes employ randomized techniques to create scatternets with certain properties. We're specifically working on reducing the number of piconets in a scatternet in order to provide various benefits, including ease of use and potential connection with other piconets. Nonetheless, minimizing the number of piconets may not always be advantageous." The maximum number of masters that can be active concurrently is at most k. It is possible to have more total device capability if there are more piconets in the scatternet. In addition, previous protocols only work when all nodes appear within a short window of time, and don't function when nodes enter and exit arbitrarily [3].

"In contrast, our scatternet forming algorithm, which we call TSF (for Tree Scatternet Formation), constructs the tree scatternet through the use of dynamic master/slave roles that are assigned to the nodes in the tree system." Both decentralized and self-healing: When nodes leave and enter the network, communication interruption time is kept to a minimum. "We chose a tree topology because it allows us to both route messages and schedule activities better with a minimum of network resources. Route planning is simplified because there are no route loops to think about. Based on our understanding, it is presumed that an ad hoc tree topology simplifies the configuration of Bluetooth networks. Trees are well suited for scheduling connections concurrently because of their hierarchical design. Concurrently scheduled ties may also exist, regardless of whether or not they are separated by stages. Remember that, when connecting to a node, it's not uncommon for customers to criticize trees as being more likely to cause disruptions." Our connection healing algorithm makes the nodes-related disruptions smaller [4].

III. TSF: TREE SCATTERNET FORMATION

TSF, a tree scatternet creation algorithm that has the following desirable properties, is introduced in this section.

- Convergence: With rapid TSF convergence, both nodes will be able to reach steady state, in which each node has access to the other. In general, whenever the TSF calculates a topology, it tries to construct a single spanning tree, and it works to converge that tree on the network with a smaller number of trees.

- Healing: TSF manages nodes that show up one at a time or in large groups, and nodes that exit one at a time or in large groups, closing loops and healing networks.

- the TSF produces topologies with the average length of node-node route being as short as possible (logarithmic in the number of nodes, avoiding long chains). TSF utilizes a randomized process when conducting data exchange and carrying out the social mission of building a more linked scatternet.

- **State Mechanics**

ISSN- 2394-5125 VOL 7, ISSUE 03, 2020

The TSF-generated scatternet is a forest comprising of linked tree components at any point in time. "Any of these trees are single nodes that tend to enter another tree in order to create a larger component and minimize the number of components, called free nodes. Both nodes other than the root node in a component are referred to as tree nodes. Each node spends a limited amount of time in each tree portion to try to rendezvous with another node belonging to another tree to ultimately form a single scatternet linked to the tree. TSF distinguishes between two kinds of part mergers in order to ensure loop-freeness: I mergers of trees each with more than one node, and ii) mergers of trees, one of which is a free node. Loops can be induced by the former, although the latter can not. TSF identifies a subset of nodes from each tree for trees of at least two nodes to be the coordinators responsible for merging with neighbouring trees. The other non-coordinator tree nodes expend a limited amount of time actively listening to question messages sent by free nodes and coordinators. Free nodes actively look for other tree or free nodes to create connection ties.

With each node working autonomously with just local contact, TSF is distributed. Three states exist: Ask, Search, and Comm. A state-machine algorithm is operated by each node in the network, switching between two of the three potential combinations of states: Inquire, Comm, and Comm/Scan. The Bluetooth Inquiry process is done by a node in the question state. A node is either idle or engaged in data contact with other nodes in its linked portion in the Comm state. In specific, free nodes stay idle in the Comm state to conserve resources. Similarly, to execute the Bluetooth Inquiry Search process, a node in the Comm/Scan state starts in the Comm state when regularly entering the Scan state [5]. Thus, a node tries to rendezvous with another node belonging to a separate tree in the Inquire and Search states to establish a Bluetooth contact channel and thus strengthen connectivity machines. The Scatternet. A node utilizes one of two separate Inquiry Access Codes when running Inquiry or Inquiry Search operations to restrict mergers to appropriate categories of nodes. Coordinators use the Restricted Inquiry Access Code (LIAC) in particular, and the Generic Inquiry Access Code is utilized for all other nodes (GIAC)." Thus, as coordinators just send and listen to LIAC, contact between coordinators is segregated from the rest of the nodes.

```
PROCEDURE TSF-FREE() {
    state_pair ← (Inquire, Comm/Scan)
    RUNFOREVER(state_pair, GIAC, 1)
}
PROCEDURE TSF-ROOT() {
    if(designated as coordinator)
        TSF-COORDINATOR()
    else
        Run(Comm, ∞, null)
}
PROCEDURE TSF-TREE() {
    if(designated as coordinator)
        TSF-COORDINATOR()
    else
        state_pair ← (Comm, Comm/Scan)
        RUNFOREVER(state_pair, GIAC, n_links)
}
PROCEDURE TSF-COORDINATOR() {
    state_pair ← (Inquire, Comm/Scan)
    RUNFOREVER(state_pair, LIAC, 1)
}
PROCEDURE RUNFOREVER(state_pair, iac, f_comm) {
    state ← random state from state_pair with 0.5 prob
    do forever
        if(state = Comm)
            t_state ← f_comm × random(E[T_inq], D)
        else
            t_state ← random(E[T_inq], D)
        Run(state, t_state, iac)
        state ← opposite state from state_pair
}
```

"Figure 2: Pseudo-code of various TSF state- machines"

ISSN- 2394-5125 VOL 7, ISSUE 03, 2020

Figure 2 displays the pseudo-code for multiple state-machines operating on different types of nodes. "When randomizing the state residence time, t-state, RUN-FOREVER initializes state arbitrarily and alternates between state pairs. As stated slightly, from relates only to tree nodes. The protocol asks the lower layers of Bluetooth to execute the related process for # percent # # # period of time on the basis of $# #. When running Inquiry or Inquiry Scan operations, the stated IAC is used as the Inquiry Access Code. To prevent periodic synchronization effects, TSF's state residence time is randomized. Two parameters depend on the randomization: E[Tinq] and D." The estimated time required to complete the Investigation process is E[Tinq]. D is a parameter that defines the duration of the random interval, which determines how long in a specified state the node remains.

As seen in the pseudo-code, in both of the two alternating states, free nodes and coordinator nodes expend about the same amount of time checking and searching for future relations. "The root nodes still reside in the state of Comm. From determines the sum of time spent in the Comm state for tree nodes, which is a feature of how active a node is likely to be in carrying out its contact activities. We estimate the optimal from value in the interest of simplicity as a function of how many connections a node has n _links [6]."

In order to maintain the invariant that when nodes enter and exit, the scatternet remains a tree, the final piece of the TSF algorithm is loop-avoidance. To complete this, only root nodes (such as during Page and Page Scan operations) are enabled to heal partitions and add a child node as a descendant. Origin, however, is in Inquiry or Search states, which are not particular about the duration of their activity. Each root designates the coordinator that is in charge of finding nearby coordinators as a node in its component tree. The component mergers are isolated from TSF, allowing root nodes to avoid having to process the extra energy-intensive work involved in the results of inquiry. We'll go into more detail about coordinator selection and part mergers in the next segment.

- **Forming Communication Links**

Nodes aim to create relations with other nodes in the Inquire and Search states. "The two nodes automatically join the Page and Page Search modes as soon as a node receives an inquiry answer successfully from another node, and attempt to create a link. The master node becomes the root when two free nodes bind, and the slave becomes a leaf node."

A single coordinator responsible for the exploration of other tree scatternets is chosen by each root node. "If only one child is the source, it elects itself as the organizer. Otherwise, when submitting a submission envelope, it selects one of the children arbitrarily and requests it to nominate the coordinator. If the selected child node is not able to become the coordinator, one of its children is automatically elected by the child node again. If a coordinator is chosen or a leaf node is hit, this method proceeds recursively. A leaf node must become a supervisor until chosen. Clearly, leaf nodes are not bottlenecks in connectivity and therefore have more surplus ability to explore adjacent computers. When a node becomes the coordinator, it sends its root to an acceptance envelope. In the Inquiry or Scan nations, coordinators look for other coordinators, as described previously." When a coordination connection is formed by two coordinators, each of them informs the corresponding root nodes with the appropriate signaling details to join the Page and Page Scan modes. "Then the coordinators sever the link between them and assume their former positions. In the meantime, the root nodes rapidly create the relation and the master node becomes the current root node and the slave becomes its child node, becoming a larger tree and reducing the forest's number of component trees. The root then automatically chooses another co-ordinator. An significant information here is that without reminding the root, the coordinator node can disappear suddenly (e.g., by crashing). We address this by restricting TOcoord slots to the life span of the coordinator position and making the root annually elect a new coordinator. This distributes the energy-intensive process of finding other coordinators equally over a vast number of nodes, in addition to rendering the protocol more stable [7].

When a relation is created by two roots, one of them assumes the combined scatternet's root function, and the other becomes a tree node. Between the Comm and Comm/Scan states, tree nodes alternate. As slaves, they can attach to free nodes. It is apparent that TSF creates loop-free topologies and only the root nodes execute part mergers. Because of space restrictions, we omit basic proof.

TSF randomizes the state residence time from an interval [E[Tinq] , D] to prevent periodic synchronization effects. D is focused on the anticipated period for two Bluetooth nodes to discover each other and create a contact connection successfully. If D is too short, the chances of establishing a link would be too poor during a slot in which there is a probability of establishing a connection." If D is too long, during slots where there is no

ISSN- 2394-5125 VOL 7, ISSUE 03, 2020

ability to create a link, a lot of time (and power) would be lost. To measure a reasonable value for D, we run simulations.

- **HealingPartitions**

For a topology training scheme, self-healing is an essential necessity, "especially in networks where certain nodes are energy-constrained (and therefore may run out of batteries) and many are mobile. We presume that network nodes can unilaterally depart," resulting in partitions of the network. Within a fair period of time, TSF guarantees the network partitions recover properly.

We discern two cases in which communication may be lost: "when a node loses its child node's link, and when a child node loses its parent's connection. It does not need to do something when a parent senses the loss of a child, except determine if it has become a free node and change its node form accordingly. If a child loses its connectivity to its parent, its node form is changed as follows. A leaf node becomes a node that is open, and a root node becomes an internal node. As stated earlier, each node proceeds to run an effective state machine."

IV. EVALUATION OF PERFORMANCE

We also created a Bluetooth simulator as an extension to the Net-work Simulator to test the efficacy of our algorithms (ns). "The bluehoc simulator built by IBM eased our development efforts. According to Bluetooth Standard Version 1.1, our simulator incorporates all significant aspects of the Bluetooth protocol stack. This provides us perspectives into the comprehension of certain technical problems as well as facets of success. In the coming weeks, we intend to release the simulator into the public domain [8]."

In two distinct conditions, we performed multiple simulations to test the efficiency of TSF: enmasse arrivals, and gradual arrivals and departures. Empirical findings on relation creation, scatternet development, and healing latencies are provided in this portion, and salient properties of the resulting topologies are addressed.

"In all the experiments, nodes are assigned to a random clock value between 0 and 2^{27}-1. Every data point shown in all figures is the average of 100 independent trials. Through simulation, we determined that the expected time to complete the inquiry process, $E[T_{inq}]$ =1.8s, when two nodes are performing opposite discovery operations namely Inquiry and Inquiry Scan. We also ran numerous simulations with two nodes running TSF to determine a good value for D. We found that setting $2.2 * E[T_{inq}] \leq D \leq D\ 4.4 * E[T_{inq}]$ would give an average connection delay of 6-8s and chose $D = 3.8 * E[T_{inq}]$ for all the simulation runs."

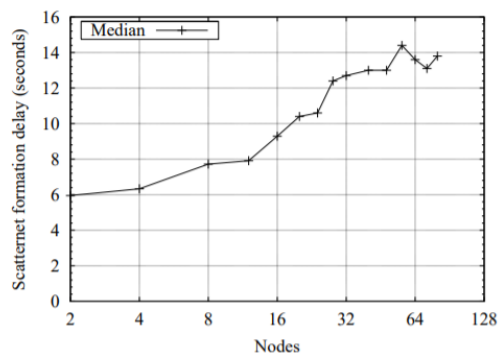
- **En masse Arrivals**

When nodes arrive en masse, we determine the TSF efficiency. In this situation, participants show up holding Bluetooth devices ad hoc and together decide on the next step. The median delays TSF needs to take to build a scatternet are shown in Figure 3. (a). That shows that as the size of the scatternet grows, the latency gets more and more proportional. a median delay of 6 seconds is required for a group of 2 nodes, and a median delay of 14 seconds is required for a group of 64 nodes. We suspect that it's because an inverse logarithmic-arithmetic average is required for TSF to calculate its scatternet creation delay. The number of components is reduced whenever a valid communication connection is established. Increasing numbers of exploring nodes increases the number of parallel connections. In other words, because every component is connected to at least one other component, each unit integrates a small portion of the total number of components.

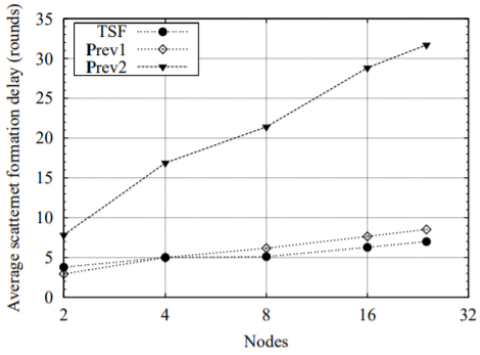
It is difficult to quantify the effectiveness of TSF's quantitative approach in comparison to previous quantitative schemes, which have been put in place in multiple simulation environments. The primary difference between the Inquiry and Inquiry Scan modes is the system's differing expectations of the efficiency of the method of forming Bluetooth connections employed by the two nodes, respectively."Schemes assume that nodes have synchronized clocks and therefore, $E[T_{inq}]$ =0.34s is dominated by the random backoff time between 0and 0.64s. We decide not to use this assumption for practical reasons and as described earlier, we found that $E[T_{inq}]$ =1.8s. We conclude that a sensible comparison between the three schemes will be to compare the scatternet for mation delay in terms of round which is simply $E[T_{inq}]$. Figure 3(b) plots the average scatternet formation delay in rounds achieved by each of the three schemes." The data points for the two previous schemes are obtained and normalized. "TSF outperforms both schemes in forming scatternets with the exception of $n = 2$. Every data point we use for represents the ideal time taken to elect the leader from nodes during Phase I as described. The

ISSN- 2394-5125 VOL 7, ISSUE 03, 2020

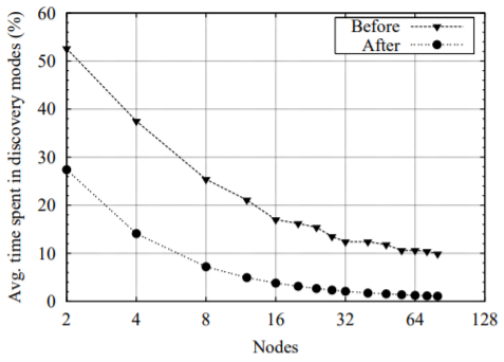
actual scatternet formation delay will be a little bit longersince the leader needs to connect to other nodes waiting in the Page Scan mode and so on. As explaine, the scatternet formation delay achieved by Prev2is longer than the other two schemes due to the synchronized nature of the algorithm. We also note that the comparison will be much more meaningful if all three schemes are developed under the same environment [9].”



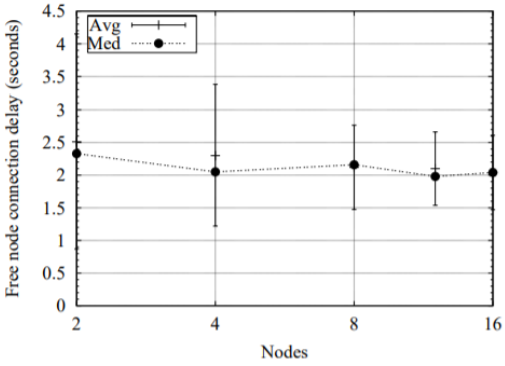
(a) Scatternet formation delay



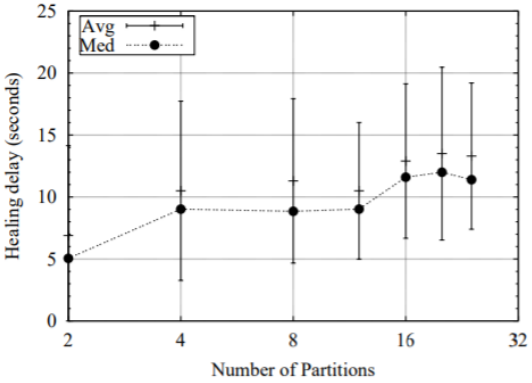
(b) Delay Comparison



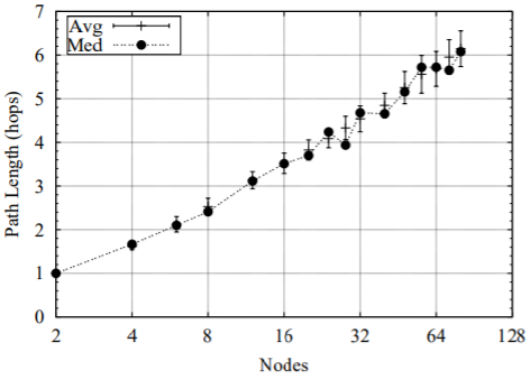
(c) Avg. time spent in Inq/Page modes



(d) Avg. free node connection delay



(e) Delay for healing network partitions



(f) Avg. path length

Figure 3: Performance of TSF in dynamic environments

We also calculated the amount of time spent by a node in finding neighbors “(Inquiry process) and creating relations with them (Page process) for two stages: before and after the creation of the linked scatternet. The percentage of time a node spends in discovery modes before and after a linked scatternet is established is seen in Figure 3(c). Not unexpectedly, with the rise in scatternet-size, the proportion reduces. As described in the Inquiry mode, the time spent dominates the overall time needed to construct a relation. In the 2-node scenario,

ISSN- 2394-5125 VOL 7, ISSUE 03, 2020

where each free node spends an equivalent amount of time switching between the states of Inquire and Comm/Scan, this is clear. In exploration modes, though, the average time spent by each node is only 52 percent of the overall time. Since only a single coordinator performs Inquiry from each part, as the scatternet-size grows, the average period a node performs Inquiry decreases. The Before curve clearly indicates that when building up a single connected scatternet, TSF enables a newly connected node to begin interacting with other nodes in its connected portion. In comparison, TSF only allows each node to invest a limited amount of time in discovery modes to bind to nodes randomly arriving after a linked scatternet is created" (less than 4 percent for scatternet-size greater than 16) [5].

- **Incremental Arrivals and Departures**

We are now testing the utility of TSF in highly diverse environments, such as supermarkets and coffee shops in malls and airports. "In these conditions, there will usually be a connected scatternet; thus, we are interested in how effectively a newly arrived node would connect to an existing scatternet and how rapidly the network heals when nodes unexpectedly leave. We note that the earlier approaches only operate when nodes enter en-masse and no nodes leave the network. We have a 32-node scatternet setup, and during a 30-second interval, nodes randomly arrive. We then measure the total link establishment latency for various numbers of arriving nodes. In relation to connecting to another free node, when the arriving nodes are spaced out over the period, each arriving free node connects to an existing non-root node in all the trials. The average link configuration delay is always less than 2.5s, as shown in Figure 3(a). The latency reduces dramatically as the number of nodes rises." This is because it becomes a bit simpler for a free node to get linked to a non-root node as the tree becomes bigger and larger.

The scatternet may be partitioned into several smaller networks when nodes spontaneously leave. In the network, "we test how quickly TSF heals partitions. As defined, coordinator nodes, one from each network partition, try to connect to each other during the healing phase. Intuitively, as the number of partitions increases, the period taken to heal the partitions of the network expands. Figure 3(e) reveals that the typical healing latency that rises logarithmically with the number of network partitions is accomplished by TSF [2].

The overall network capacity and average latency for any two nodes are influenced by a Bluetooth scatternet topology. In multi-hop networks, the path length or hop count between communicating nodes greatly influences the end-to-end latency." Figure 3(f) shows that with the sum of nodes contained in the scatternet, the cumulative path length increases logarithmically.

V. DISCUSSION AND CONCLUSION

TSF is designed for use in instances where high-bandwidth network connectivity and node organization are more important than achieving maximum network throughput. "It enables nodes to show up and leave whenever they like, building a tree topology layer by layer and repairing split-levels when they form. In addition, a new arrival node can immediately start communicating with the nodes in its connected component once it has arrived. TSF is part of a larger scheduling strategy that serves as a tool for managing dynamic environments with self-organizing scatternets. We've simulated the tree scatternet formation and found that it occurs exponentially with the number of nodes. It is worth noting that while requiring each node to spend just a small amount of time discovering neighbors, TSF achieves low average link establishment delay. In the case of large networks, the path length between two nodes is approximately a logarithmic function of the network size. TSF can, but may not, be able to produce a connected scatternet in the event that nodes are within radio proximity." TSF places restrictions on the activities of partition discovery and merging. "When coordinators or roots cannot hear each other, the algorithm fails to produce a connected scatternet. TSF is applicable to networks that have a diameter greater than one. When there is a connected physical topology, a companion loop-detection protocol can be created to ensure the existence of a tree scatternet."

"We will investigate the self-stabilization procedure for constructing the scatternet in a dynamic wireless sensor network, where sensor nodes may fail due to power failure or environmental impact, or sensor nodes may be relocated."

REFERENCES: -

[1] Parra, Lorena & Marín, José & Mauri, Pedro & Lloret, Jaime & Torices, Virginia & Masaguer, Alberto. (2019). Scatternet Formation Protocol for Environmental Monitoring in a Smart Garden. Network Protocols and Algorithms. 10. 63. 10.5296/npa.v10i3.14122.

ISSN- 2394-5125 VOL 7, ISSUE 03, 2020

[2] Sharafeddine, Sanaa & Al-Kassem, Ibrahim & Dawy, Zaher. (2012). A scatternet formation algorithm for Bluetooth networks with a non-uniform distribution of devices. J. Network and Computer Applications. 35. 644-656. 10.1016/j.jnca.2011.10.004.

[3] Liu, X. & Al-Anbuky, Adnan. (2009). Bluetooth Information Exchange Network. Proceedings of the 2008 Australasian Telecommunication Networks and Applications Conference, ATNAC 2008. 169 - 174. 10.1109/ATNAC.2008.4783317.

[4] Nieto, Juan & Candolfi, Norma & Michel-Macarty, José & Jimenez-Garcia, Elitania. (2009). Bluetooth Performance Analysis in Wireless Personal Area Networks. Electronics, Robotics and Automotive Mechanics Conference. 38-43. 10.1109/CERMA.2009.48.

[5] Jung, Sewook & Gerla, Mario & Kalló, Csaba & Brunato, Mauro. (2005). Decentralized Optimization of Dynamic Bluetooth Scatternets.. 305-313. 10.1109/MOBIQUITOUS.2005.21.

[6] Persson, Karl & Manivannan, D. & Singhal, Manish. (2005). Bluetooth scatternets: Criteria, models and classification. Ad Hoc Networks. 3. 777-794. 10.1016/j.adhoc.2004.03.014.

[7] Melodia, Tommaso & Cuomo, F.. (2004). Locally Optimal Scatternet Topologies for Bluetooth Ad Hoc Networks. 2928. 116-129. 10.1007/978-3-540-24614-5_9.

[8] Zhang, Chu & Wong, V.W.S.. (2004). TPSF+: a new two-phase scatternet formation algorithm for Bluetooth ad hoc networks. 6. 3599 - 3603 Vol.6. 10.1109/GLOCOM.2004.1379037.

[9] "G. Tan and J. Guttag. A Locally Coordinated Scatternet Scheduling.""Algorithm. In The 27th Annual IEEE Conference on Local Computer Networks (LCN), Tampa, FL, Nov. 2002."