

A Bluetooth based Sensor Network for Civil Infrastructure Health Monitoring¹

Vipin Mehta

School of Information and Computer Science
University of California, Irvine
Irvine, CA - 92697
vipin@ics.uci.edu

Magda El Zarki

School of Information and Computer Science
And CAL-(IT)²
University of California, Irvine
Irvine, CA - 92697
elzarki@uci.edu

Abstract — Communicating with sensors has long been limited either to wired connections or to proprietary wireless communication protocols. Using a ubiquitous and inexpensive wireless communication technology to create Sensor Area Networks (SANs) will accelerate the extensive deployment of sensor technology. Bluetooth, an emerging, worldwide standard for inexpensive, local wireless communication is a viable choice for SANs because of its inherent support for some of the important requirements - low power, small form factor, low cost and sufficient communication range.

In this paper we outline an approach, centered on the Bluetooth technology, to support a sensor network composed of fixed wireless sensors for health monitoring of highways, bridges and other civil infrastructures. We present a topology formation scheme that not only takes into account the traffic generated by different sensors but also the associated link strengths, buffer capacities and energy availability. The algorithm makes no particular assumptions as to the placement of nodes, and not all nodes need to be in radio proximity of each other. The output is a tree shaped scatternet rooted at the sensor hub (data logger) that is balanced in terms of traffic carried on each of the links. We also analyze the scheduling, routing and healing aspects of the resulting sensor-net topology.

Keywords – *Bluetooth, scatternet, topology design, load balancing, simulated annealing, link strength, scheduling, timing accuracy, power consumption.*

1 INTRODUCTION

Sensor Area Networks (SANs) will play an important role in the deployment of next-generation, health-monitoring systems for highways, bridges and civil infrastructures. In such a framework, data from thousands of sensors will be analyzed with long-term and real-time assessment decision-making implications. A flexible and scalable architecture will be needed to integrate real-time heterogeneous sensor data, database and archiving systems, and rational statistical decision-making procedures. Applications include long-term condition assessment and emergency response after natural or man-made disasters and acts of terrorism, for all types of large constructed facilities.

Recent advances in micro electromechanical systems (MEMS) technology have resulted in cheap and portable devices with integrated sensing, computing and communication capabilities. A network of these devices can be used for automated information gathering and distributed micro sensing in many civil applications such as home energy management, civil infrastructure health monitoring, etc. For the latter application in particular, Civil Engineers need the flexibility to place sensors in locations that are critical for health monitoring but that may not be convenient for existing wiring schemes. In many situations it is difficult to rewire the existing infrastructure. One possibility is to use power line

¹ This work is supported in part by NSF ITR grant # 0205720 and a fellowship for the Center for Pervasive Communications at UC, Irvine, CA.

communications to provide the required network connectivity. This introduces additional complexity and constraints in terms of scalability and the available throughput. Powerline networking has not been broadly accepted and is currently only being offered as a home networking solution. In addition the powerlines may not give adequate coverage either. Many of these scenarios thus call for wireless sensor networks as opposed to wired ones. Wireless networks can be much more cost and time effective and are also easier to deploy especially in remote locations. In some application scenarios, a wireless solution can vastly reduce the monitoring installation cost, where the cabling alone generally constitutes 30-45% of the total cost.

In this paper we address the problem of designing a SAN composed of fixed wireless sensors, for deployment in *existing* civil infrastructures such as bridges, and highways, to monitor stress, vibration, temperature, humidity, etc. Although the sensors are fixed, they are deployed in an ad hoc fashion (dependent upon need and accessibility) and sensors can die and be replaced, or new ones added, at any time. Given current wireless networking technologies, the specific needs of such sensor networks and the desire for a near term deployment, we propose a solution based on the Bluetooth standard. In section 2 we highlight the networking issues and address the suitability of using Bluetooth technology for this specific application. In section 3, we discuss our proposed solution. Section 4 presents some of the important aspects of the topology – scheduling, routing, healing, delay, data dissemination and timing accuracy. In Section 5, some results, related to the comparison of topology selection phases between two different approaches, have been discussed. We conclude in section 6 with some observations and future prospects

2 BLUETOOTH TECHNOLOGY

Although Bluetooth was not originally conceived to be used for these purposes, there is good overlap of the features supported by it (addressed in detail below) and the requirements for fixed large sensor networks. The only key existing technologies in this space are the Berkeley notes [19] and the upcoming IEEE 802.15.4 standard [20]. There is an ongoing effort to develop networking protocols that will allow the deployment of Berkeley notes in larger numbers. Extensive studies need to be conducted on their battery life, early tests show a longevity in the range of 3-4 days, some claim 3 months at a extremely low sampling rates. The application we are addressing requires continuous sampling, e.g., accelerometers measuring vehicle speeds on a bridge. The IEEE 802.15.4 standard is not designed for sensor networks and as such does not address some of the unique features of this type of application. Today, with a

broader specification and a renewed concentration on interoperability, manufacturers are ready to forge ahead and take Bluetooth products to the marketplace. A single chip embedded Bluetooth design for such sensors can meet the growing demand for connected information appliances where cost and flexibility are the key features for massive deployment. The authors are aware that the deployment of Bluetooth nodes in large numbers in a scatternet formation has not yet been fully tested but the basic Bluetooth nodes are readily available and the specifications for networking are well laid out.

Generally, a wireless sensor network is energy-constrained, has low bandwidth, and contains hundreds to thousands of sensors. Bluetooth, with its inherent support for low power modes, small form factor and minimal associated costs, emerges as a natural choice for satisfying the requirements of a fixed SAN. A list of features provided by Bluetooth relevant to wireless sensor networks can be summarized as follows:

1. Low cost (sub \$5 target price)
2. Low power (100m at +20 dBm with Class 1 radio operation, +4 dBm power with Class 2 radio operation, 10m at 0 dBm with Class 3 operation)
3. Small form factor (Highly Integrated ASICs, e.g.: Motorola BTMCM150 Bluetooth module containing all the required hardware and software components in a single 15 x 20 x 3 mm package).
4. Frequency Hopping, Time Division Duplex (FH-TDD) scheme using the ISM band (2.4 - 2.483 GHz range) to minimize the scope for interference and allowing non line of sight communication.
5. Power saving modes - Park, Hold, Sniff, Standby.
6. Asymmetric Data rates allowing the provision to allocate an uplink bandwidth (from slave to master) of 723.2 Kbps and downlink bandwidth of 57.6 Kbps.
7. Support for both Asynchronous Connection Less (ACL) and Synchronous Connection Oriented (SCO) (can be used for transmitting voice/audio from the sensor node itself) channels.

Bluetooth is a frequency hopping (FH) system that defines multiple channels for communication (each channel defined by a different frequency hopping sequence). A group of devices sharing a common channel constitute a *piconet*. Each piconet has a master unit that selects a frequency hopping sequence for the piconet and controls the access to the channel. Other participants of

the group, known as slave units, are synchronized to the hopping sequence of the piconet master. Within a piconet, the channel is shared using a slotted time division duplex (TDD) protocol where a master uses a polling style protocol to allocate time-slots to slave nodes. The maximum number of slaves that can simultaneously be active in a piconet is seven. Multiple piconets can co-exist in a common area because each piconet uses a different hopping sequence. Piconets can also be interconnected via bridge nodes to form a bigger ad hoc network known as a *scatternet*. A bridge node can be a master in one piconet and a slave in others (termed as an M/S bridge) or a slave in all the piconets (termed as an S/S bridge).

3 AN INTEGRATED SOLUTION

An integrated solution can be realized, wherein using the Smart Transducer Interface Module (STIM) specifications as described by the IEEE-P1451.2 standard, the module can be directly integrated with any available off the shelf Bluetooth module to achieve a wireless sensor node [2]. A cluster of such nodes can then be tied together using a Bluetooth scatternet topology to form a fully functional SAN. Besides providing a flexible wireless transport medium, such an arrangement will also allow for:

- i. *Scalability* to include any number of sensors at any point of time,
- ii. *Robustness* against link breakdowns and node failures, and
- iii. *Self Healing*, re-configuration of the system.

Every such scatternet formation shall be rooted at a sensor hub that will act both as a local logger for the system and as an arbitrator during the knowledge discovery and topology formation phase (described in the sections below). It acts as an interface between the SAN and the Internet infrastructure and can be realized via any of the available technologies, such as 802.3 or 802.11. This sensor hub is called the Super Master (SM)² and is responsible for the formation of the scatternet. The SM is responsible for:

- i. Topology or scatternet formation
- ii. Scheduling (access and bandwidth allocation)
- iii. Routing (two way communication capability)
- iv. Healing (connectivity maintenance)

² We shall use the words sensor hub and SM interchangeably throughout the text.

Because of its importance in the centralized scheme we are proposing here, it is assumed that the Super Master is fully replicated to provide fault tolerance.

3.1 Scatternet Formation

A number of algorithms have been proposed to form a sensor network topology. In [10], the author computes an energy efficient sub-network, namely the MECN, when a communication network is given. SMECN is proposed by [11] to also provide such a sub-network. Similarly extensive literature can be found on routing techniques from conventional flooding to more sophisticated schemes such as gossiping [17], SPIN [18], SAR [12], LEACH [13] and Directed diffusion [14]. Similarly, a number of MAC protocols have been proposed both for fixed allocation and random access mechanisms [12, 15], the SMACKS and EAR protocol [12], CSMA based [15] and a hybrid TDMA/FDMA scheme [16]. Although these different protocols adapt well to the underlying application requirements, their main disadvantage is that they operate in a non-standardized and often proprietary infrastructure, which makes their deployment and interoperability a cause of concern.

We propose a topology formation scheme for a wireless sensor network based on Bluetooth technology, the radio and MAC for which has already been standardized and thoroughly tested. We also introduce a zero overhead routing mechanism that communicates and routes the data to and from the sensor nodes on the fly. The SM is responsible for creating a neighbor node matrix, which it then uses to create the scatternet tree. The entries in the neighbor node matrix consists of data collected by each node from their neighboring nodes. The entire approach is centralized in nature primarily because it not only allows for a more optimum network topology generation using global information, available at the sensor hub, but also makes the scheduling, bandwidth allocation and routing of the traffic easier. In addition, the individual sensor nodes might not be endowed with a sufficiently high amount of processing and battery power to be able to take on such a load. The SM thus executes the complete algorithm while the rest of the nodes just act on the decisions taken by the sensor hub. The process can be divided into two phases, namely:

- a) Knowledge Discovery phase, and
- b) Connection Setup phase

Before proceeding with a description of the aforementioned stages and their role in the topology formation of an SAN, we would like to introduce the following terms:

- i. *Load Factor*: A measure of the average amount of traffic generated by a particular node over a period of time.
- ii. *Degree*: Number of nodes in radio proximity to a sensor node (i.e. also referred to as neighboring nodes).
- iii. *Buffer Capacity*: The amount of storage space available in a sensor node for storing its own sensor data and the sensor data generated by its slaves, as well as the data collected, during the Knowledge Discovery phase, by the sensor from its neighboring nodes, defining their capabilities.
- iv. *Link Strength*: The strength of the radio link between two neighboring sensor nodes.
- v. *Energy*: The amount of energy available to the node from the energy source at any time.
- vi. *Clock Value*: Refers to the native clock value of a node responding to an inquiry message. The value is exchanged during the Knowledge Discovery phase and is used to expedite the connection setup between two sensor nodes during the Connection Setup Phase.

3.1.1 Knowledge Discovery Phase

The Knowledge Discovery phase is characterized by having each node collect information about all the other nodes that are in radio proximity to itself and ultimately communicate this data to the SM to create the neighborhood matrix. During this phase each node, in turn, queries its neighbors during an *Inquiry phase*, in which the node itself is in Inquiry mode and all the other nodes are in Inquiry Scan mode. In contrast to the approaches proposed by Salonidis et al. (symmetric link formation) [3] and Law et al. (randomized distributed scatternet formation) [4] only one node at a time is allowed to be in the Inquiry mode and the rest of the nodes are in the Inquiry Scan mode. A node will determine the end of *its* Inquiry phase when it does not hear from any new neighbors for a given interval. The 'baton' (i.e., the right to start an Inquiry phase, or in other words be in Inquiry mode) is then passed onto another node that has not yet initiated its own Inquiry phase to collect neighbor information. The time duration of the Knowledge Discovery phase will be a function of network density. This particular arrangement has the benefits of extracting the maximum possible information about the interconnectivity among different nodes that will be used subsequently during the Connection Setup phase when masters have to select a maximum of k slaves out of several possible neighbor nodes. The algorithm employs a *backtracking* methodology to traverse the

entire set of nodes recursively in a Depth First Search (DFS) fashion till it hits the terminating condition.

The Knowledge Discovery phase is initiated by the sensor hub that goes into Inquiry mode thus starting the first Inquiry Phase. Henceforth we will refer to a node that is in Inquiry mode as the *inquiry* node. The process can be summarized as follows:

- i. **Inquiry phase**: Every node, including the SM, maintains a database of one hop away nodes, i.e., nodes that are in radio proximity. Whenever an inquiry node gets a response to its inquiry message, it checks its database to verify if an entry for this neighbor node already exists. If no entry is found, it increases its connectivity counter, *degree*, by one, and establishes a connection with the neighbor to exchange load factor, buffer capacity, and the corresponding link strength information using the Bluetooth defined Service Discovery Application Profile (SDAP) primitives. When no new nodes are discovered in a specified interval, the inquiry node terminates its Inquiry phase.
- ii. **Passing the baton**: The inquiry node then passes the baton to one of the neighbor nodes that it discovered during its Inquiry phase. The 'new' inquiry node repeats the same procedure as in step (i). After passing the baton, the 'old' inquiry node does not go back into Inquiry Scan mode; it enters a stand by state in which it does not respond to Inquiry messages. Because of the symmetric nature of the neighbor node matrix, the SM can extrapolate neighbor information without having each node collect reverse information. This will reduce the traffic load and shorten the discovery period. Only nodes that have not experienced an Inquiry phase are in Inquiry Scan mode.
- iii. **Terminating**: The terminating condition along any branch is satisfied when the current inquiry node has no other node to pass the baton to. This happens when no neighbor responds to its inquiry packet thus it has no entries in its database. The current inquiry node then informs its 'parent', i.e., the node that passed the baton to it, that the end has been reached. The parent then either picks another neighbor node from its database to pass the baton too, or, if no other candidate neighbor (i.e., node that has not yet had its Inquiry phase) is found in its database, it signals in turn its parent, which repeats the process. This backtracking technique is used until no new neighbors are found that have not yet been in the Inquiry phase. With each

backtrack step, the child node passes all its database information to the parent node which in turn stores it and then passes it along to its parent when it comes time for it to backtrack to the next level up.

At the end of the terminating process, all nodes have passed their database contents up the tree to the SM. The SM then proceeds to create the neighbor node matrix by using all the collected data. The Knowledge Discovery Phase enables the SM to determine the following characteristics about the sensor nodes participating in the scatternet:

- i. Bluetooth device address
- ii. Clock values that will be used to expedite the connection setup phase
- iii. Number of one hop neighbors for all the nodes (also referred to as their *degree*)
- iv. Link strength of those one-hop neighbors
- v. Their load factors
- vi. Amount of energy available to them
- vii. Their buffer capacities

3.1.2 Connection Setup Phase

This phase is entirely orchestrated by the SM using the information gathered during the Knowledge Discovery phase. The SM starts by selecting k slaves from its list of one-hop neighbors. The choice of these k slaves, to yield a balanced configuration, becomes increasingly difficult to pick as we go down the hierarchy and also as the network density increases. After the slaves are selected, they act as masters for the next level of the tree hierarchy but the process is still coordinated by the SM only. The entire algorithm proceeds in a Breadth First Search (BFS) fashion in contrast to the DFS nature of the Knowledge Discovery phase.

The slave nodes are selected based upon the k highest *Slave Admissibility Factors (SAF)s*. The SAF determines the suitability of one slave over another while being assigned to a particular master. This governs the selection of slaves out of the one-hop neighbor pool whose number may be considerably larger than k . For any node, the important factors contributing towards the determination of SAF are:

- i. Strength of the link between the master and the slave (Link Strength).
- ii. Number of one-hop neighbors to the master or its (Degree).

- iii. Amount of energy available or remaining with the node (Energy)
- iv. Amount of traffic generated by the nodes (Load Factor).

Below, we discuss each one of the factors in detail and how they figure in the expression of the SAF (Φ).

Link Strength (Λ): The weight contributed by link strength is directly proportional to its value above a pre-determined threshold. i.e.:

$$\Phi \propto (\Lambda - \Lambda_T) \quad (1)$$

where Λ_T = Threshold Link Strength

A good estimate of link strength can be obtained using the *Get_Link_Quality* primitive which is implementation dependent and can be based on Signal to Noise ration calculations at the receiver corresponding to each of the senders assuming that they are transmitting at the same power. Using the link strength serves a dual purpose. Firstly it makes sense to filter out the nodes with lower link strength. Secondly, the threshold also allows us to reduce the likelihood of intermittent link loss and high error rates by selecting slaves with sufficiently high link strengths.

Degree (θ): The weight contributed by the degree counter is inversely proportional to its adjusted value minus one, i.e.:

$$\Phi \propto (\theta - \theta_A - 1)^{-1} \quad (2)$$

where θ_A , an ‘adjustment’ parameter, takes into account the neighboring nodes that have already been selected as a part of the topology in the upper levels of the hierarchy.

This ensures that a node having a degree of only one at any stage has the highest possible likelihood to be selected as a slave. If unselected, it will be left as an orphan. Therefore no node or cluster will emerge as an independent entity (disconnected) so long as it is in radio proximity of any of the nodes in the main tree.

Energy (ε): The weight contributed by the available energy is directly proportional to its value. i.e.:

$$\Phi \propto \varepsilon \quad (3)$$

The notion of ‘available energy’ playing a role in the topology determination is particularly important because different nodes may be having varied energy levels (possibly because of their spatial orientation in the case of solar powered nodes). And since the nodes higher up in the hierarchy expend considerable amount of energy in routing the network traffic of other nodes, they need to be positioned appropriately.

Load Factor (λ): The weight contributed by the load factor is directly proportional to its value. i.e.:

$$\Phi \propto \lambda \quad (4)$$

This allows the algorithm to capture the nodes with the highest load first as they have the potential of creating maximum imbalance if allowed to percolate down the hierarchy (e.g., a single leaf with a heavy load on a branch). Another reason for using this strategy is the reduced energy consumption of the sensor network as will be shown later in section V. Intuitively, the traffic from the nodes at the lower levels of the hierarchy take progressively larger number of hops to reach the SM. Hence if the nodes with high load factors are placed in the lower levels then the traffic from them has to cover more distance thus expending a considerably higher amount of energy just for routing the traffic.

The SAF can thus be summarized using the following equation:

$$\Phi = [(\Lambda - \Lambda_T) / \Lambda_M]^\alpha \cdot [(\theta - \theta_A - 1) / (\theta_M - 2)]^{-\beta} \cdot [\varepsilon / \varepsilon_M]^\gamma \cdot [\lambda / \lambda_M]^\delta \quad (5)$$

where Λ_M , θ_M , ε_M , λ_M are maximum link strength, maximum number of sensor nodes, initial energy and maximum load factor, respectively and α , β , γ and δ are tunable parameters. Each of these factors can, therefore, be made to vary their contribution by tuning their respective exponents. This gives the application a significant control over the criteria for selecting the topology and which of the factors should contribute more weight.

Besides SAF, load balancing is also an important criterion in selecting the slaves at any point in the hierarchy. In a practical situation we will have a number of different types of sensors with varying load factors. There should be an eclectic mix of their placement in the hierarchy to avoid their clustering in a single piconet that might unduly load one particular branch of the tree thus creating a potential threat to the stability of the link or the node. With similar amount of traffic on the different links at the same level in the hierarchy, the energy gets drained almost equally from the siblings thus allowing a graceful degradation of the sensor network services. We use a novel computation technique to realize this. We consider the selection of nodes based on the highest SAFs, at the same time optimizing their placement for load balancing. Since the number of possibilities of assigning slaves to a master increases exponentially with the number of slaves, especially when the network density is high, the selection problem can be formulated as a combinatorial optimization problem and can thus be solved by using an evolutionary computation technique. We have adopted Simulated Annealing (SA)[9] for this purpose and we compare it against another approach called Greedy Search (GS).

We begin by defining the terms that will be used to describe both the algorithms:

- i. *Element*: Any one-hop neighbor to the node being considered as master.
- ii. *Domain*: A set consisting of a cluster of one-hop neighbor elements headed by a master.

$$D_i^h = \{m_i^h, \{s_{ij}^h\}_{j=1 \dots n_s}\},$$

where h indicates the level of hierarchy, i indicates the i^{th} domain in the hierarchy, m refers to the master of the domain, s refers to the one hop neighbor elements to the master, n_s refers to the number of such elements.

- iii. *Configuration*: A collection of several domains headed by their masters. Consists of a variable and a fixed portion. The fixed portion is contributed by the slaves that were selected in the previous configuration and are now going to act as masters in the current hierarchy level. The variable portion is constituted by the domains corresponding to each of these masters, which can keep changing as slaves are transferred from one domain to another to generate random configurations.

$$C^h = \{D_i^h\}_{i=1 \dots n_m^h},$$

where n_m^h denotes the number of domains (or masters) in a configuration.

- iv. *Covalency*: The number of masters that share a particular element in the current configuration is the covalency of that element. It is basically the number of domains an element can be a part of.

3.1.2.1 Simulated Annealing

We now describe the SA process in detail. A typical SA process can be defined using the following four components:

- i. *Root Configuration*: A base level configuration is generated by randomly distributing the elements among different domains.
- ii. *Rearrangements*: An efficient generation of successive configurations will increase the rate of convergence of the solution and also the quality of the solution itself. We resort to a process called '*Tossing of Slave*' which follows an adaptive probability distribution profile.

Ideally, a node should not have any preference for being an element in any of the domains in a given configuration. But

simulations show that if they are allowed to select any domain with a uniform probability distribution profile then some of the masters might end up having more than k elements while some have to be content with a number substantially less than k . The SA algorithm in such cases is not able to balance the load factor effectively as the bottleneck is created by the domain having the least number of elements. To counter this effect, the probability that an element will fall under any master is proportional to the deficiency of that particular master to reach an element count of $k+1$. Also, the ones that have already reached the count of k are considered to saturate at that number irrespective of the number of elements they have in excess to k . This enables a master with a current element count of k' to attract an element with a probability $p \propto (k + 1 - k')$. The advantage of such a distribution is that although it enables the elements to prefer the masters having a lesser number of elements, it also lets the masters that have reached an element count of k to have a finite probability of attracting different elements and thus allowing the flexibility that the SA algorithm needs to generate sufficiently random configurations and select the best out of them. Also, instead of holding the temperature for a fixed number of configuration changes as in a typical SA process we allow it to vary so that each of the elements having a covalency of more than one is able to contribute its part in converging to a configuration with minimum energy.

- iii. *Objective function:* The objective function is chosen to minimize the load variation across different piconets at a particular level of the hierarchy. Considering its aggregate effect, one can clearly see that such an approach would balance the bandwidth distribution across all the links for any master at any particular level in the topology. A possible choice that we used is the variance of the load distribution. Another choice that could be equally effective would be to minimize the difference between the maximum and minimum load factor.

A piconet is a subset of the domain ($P_i^h \subseteq D_i^h$) and can be defined as

$$P_i^h = \{m_i^h, \{s_{ij}^h\}_{j=1 \dots \min(n_s, k)}\},$$

where the range of j includes $\min(n_s, k)$ number of elements with highest SAFs. If ω_i^h denotes

the total bandwidth generated by piconet P_i^h and λ_{ij}^h denotes the load factor of the element s_{ij}^h , then

$$\omega_i^h = \sum_{j=1}^{\min(n_s, k)} \lambda_{ij}^h \quad \text{and} \quad \bar{\omega}^h = \frac{1}{n_m^h} \sum_{i=1}^{n_m^h} \omega_i^h, \quad (6)$$

where $\bar{\omega}^h$ is the average load factor. The energy of a configuration can thus be defined as

$$E^h = \omega_\sigma^h = \frac{1}{n_m^h} \sum_{i=1}^{n_m^h} (\omega_i^h - \bar{\omega})^2 \quad (7)$$

- iv. *Annealing Schedule:* The annealing schedule is selected so that it is a monotonically decreasing function of temperature. The maximum and minimum values are chosen keeping in mind the range of ΔE values encountered during the crystallization phase

$$T_{i+1} = T_i \times \eta, \quad \text{where } \eta < 1 \quad (8)$$

The algorithm starts with the SM selecting k of its slaves based on the highest SAFs. The annealing process is then applied to each level of the hierarchy to converge to the best possible configuration for that level. The nodes that could not be selected are set free and the process is repeated for every level. Every level of the hierarchy thus undergoes a crystallization phase to generate the network topology, level by level. The algorithm can briefly be summarized as follows:

```

Procedure Create Topology()
{
    nodes left = Max Sensor Nodes
    while (nodes left)
    {
        configuration ← Crystallize()
        nodes left ← Update the
                        topology(configuration)
    }
}

Procedure Crystallize()
{
    configuration ← Generate a random
                    config(nodes left)
    temperature = Tmax
    while (temperature > Tmin)
    {
        while (repeat)
        {
            repeat ← Modify the
                    config(configuration)
            energy ← Evaluate the
                    config(configuration)
        }
    }
}

```

```

 $\Delta E \leftarrow \text{energy}_{(\text{current configuration})} - \text{energy}_{(\text{last accepted configuration})}$ 
acceptable  $\leftarrow$  Metropolis( $\Delta E$ , temperature)
if (!acceptable)
{
    configuration  $\leftarrow$  Restore the earlier config()
}
}
temperature  $\leftarrow$  Annealing Schedule()
}
return configuration
}

```

Lets take a snap shot view of the algorithm at a particular level of the hierarchy. It starts by generating a random configuration that assigns the elements randomly to the available master domains. The algorithm then modifies the configuration by tossing one element at a time from the domain of one master to that of another. For every domain, the slaves are selected based upon their SAFs. The configuration is then evaluated by summing up the bandwidth contributed by each of the domains or piconets (selected slaves + master). The energy of the entire configuration is determined by calculating the variance of the load distribution obtained above. The change in energy is calculated with reference to the previous accepted configuration. The ΔE is then fed into the *Metropolis function* ($e^{-\Delta E/T}$), which decides on the acceptability of the configuration based on temperature and the difference in the amount of energy. This modification-evaluation phase is continued till the temperature falls below a threshold.

3.1.2.2 Greedy Search

The basic formulation of the problem remains the same. We generate the network topology level by level. At each level (except the last) the slaves of the previous hierarchy act as masters in the current one. A *root configuration* is generated by randomly distributing the elements among different domains. The *rearrangements* to the configuration are applied as follows. An element having a covalency of more than one is selected and is repeatedly made to fall in all the domains it can be a part of. The process is repeated for every element within a configuration. The algorithm thus operates in a two-tier fashion. In the first stage a particular element from the current configuration is selected. In the second stage the slave is made to take the membership value of every domain to which it can belong. The *object function* remains the same as in (7). A significant difference between SA and GS is the acceptance criteria, which is simply based on the change in energy. If its negative then the configuration is accepted otherwise it is rejected. The algorithm can be summarized as under:

```

Procedure Create Topology()
{
    nodes left = Max Sensor Nodes
    while(nodes left)
    {
        configuration  $\leftarrow$  GSearch()
        nodes left  $\leftarrow$  Update the topology(configuration);
    }
}

Procedure GSearch()
{
    configuration  $\leftarrow$  Generate a random config (nodes left)
    num_elements  $\leftarrow$  Get number of elements(configuration)
    while (num_elements)
    {
        element  $\leftarrow$  Select a new element (configuration)
        iterations  $\leftarrow$  Get covalency (element)
        while (iterations)
        {
            repeat  $\leftarrow$  Modify the config(configuration)
            energy  $\leftarrow$  Evaluate the config(configuration)
             $\Delta E \leftarrow \text{energy}_{(\text{current configuration})} - \text{energy}_{(\text{last accepted configuration})}$ 
            acceptable  $\leftarrow$  ( $\Delta E < 0$ ) ? TRUE : FALSE
            if (!acceptable)
            {
                configuration  $\leftarrow$  Restore the earlier config ()
            }
            iterations -
        }
        num_elements -
    }
    return configuration
}

```

Once the entire network topology is determined, the SM instructs the nodes to start the actual connection process by communicating the topology information down the tree hierarchy. The important issue here is the time complexity of the scatternet formation phase which can be on the high side. But since the sensor nodes are fixed in their behavioral characteristics in the sense that they do not move, so the time cost incurred is worth establishing a network that is optimized on all of the following fronts - link strength, cluster connectivity (i.e., any node that is in radio proximity of any of the other

node should automatically fall into the tree hierarchy rooted at the SM) and balanced loading.

4 TOPOLOGY ANALYSIS

This section covers some of the important aspects relevant to the sensor networks and analysis of the same in the context of the proposed topology.

4.1 Scheduling

A tree topology simplifies the scheduling of the communication links in the resulting sensor network. The hierarchical nature of a tree allows links at the same level to be scheduled in a time division multiplexed fashion without conflicts. Furthermore, links at non-consecutive levels can also operate simultaneously resulting in an alternate layer functioning topology thereby increasing the throughput of the overall system. Consider a single piconet in the scatternet. The master of the piconet polls the slaves with a TDD schedule. It collects the data from all these nodes one by one along with the sensor data each one of them collected from the tree below during the previous epoch. Once it has gathered the data, it puts the slave nodes in *Sniff* mode [1] till a time duration ‘ T_s ’ is exhausted. This operation is performed in parallel by all the piconets present at the same level in the hierarchy. At the end of ‘ T_s ’ the master puts the slaves in *Hold* mode [1] for the same duration ‘ T_s ’ to allow them to act as masters in their own respective piconets. At the end of ‘ T_s ’ the current masters put their slaves into Hold mode and return to their original masters as slaves to report the data. This completes one phase of data collection and reporting. The process thus repeats with a periodicity of $2 * T_s$. A good choice of ‘ T_s ’ is guided by the amount of data individual nodes can buffer and the data incident at those nodes. Thus,

$$T_s < \min_{i=1, \dots, N} \{B_i / C_i\} \quad (9)$$

where ‘ B_i ’ is the buffer capacity at node ‘ i ’ and ‘ C_i ’ denotes the net throughput at that node, ‘ N ’ being the total number of nodes. This will avoid buffer overflows and the resulting loss of data associated with it.

4.2 Routing and Addressing

A tree topology simplifies routing because of some of its inherent properties.

- i. no routing loops can exist.
- ii. there exists a unique path between any node and the sensor hub. Nodes can be assigned unique addresses based upon their position in the tree. A higher-layer destination identifier (eg IP address of the sensor hub) in conjunction with the unique node address can be used to address any

particular node from the network management perspective.

Using these properties, we propose a novel addressing scheme that not only reduces the overhead associated with carrying the Bluetooth Device addresses (48 bits) all the way but also avoids maintaining tables or complex routing information anywhere in the sensor network. The latter is of significant importance as the nodes have a very limited amount of memory. Since there is an additional cost involved in transmitting every extra bit of information when the SM has to address a specific node in the sensor network, the scheme reduces the length of the addressing portion to just 24 bits and also avoids the overhead of the message exchanges normally associated with collecting and collating the routing information. The inquiry packet header has the following structure:

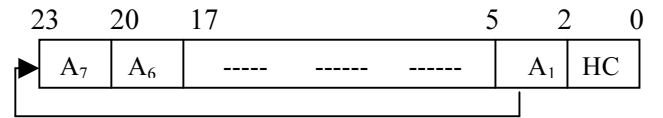


Figure 1. Query Packet Header

The three least significant bits refer to the hop count from the SM, the SM being at level zero. When the sensor hub has to address any node in particular, it fills up the ‘hop count’ field equal to the hierarchy level at which the node lies. At every hop the ‘hop count’ field gets decremented by one so that the packet is delivered to the piconet when the hop count reaches a value of zero. Every three bits, starting from the position immediately after the hop count field, is used to select a node within the current piconet. Since there can only be seven slaves to a given master, the three bits are sufficient to uniquely identify a node within the piconet. The convention followed here is that the network addresses are assigned in the increasing order of the corresponding physical device address. The all ones address is reserved for broadcasting in the piconet. This is helpful particularly in region based addressing as will be discussed later. Starting from the first bit at the end of the ‘hop count’ field, a block of three bits is rotated in a clockwise direction at every hop. This allows a master at any level to obtain its piconet specific address from the first address field directly rather than actually going to an offset that would have contained the address if the shifting operation were not done at every hop. The added advantage is that we do not need any offset information. The SM generates a table based on the above scheme during the topology formation phase. At every step of crystallization, the SM assigns the addresses of the nodes it has decided upon and updates its Bluetooth Device address to network address mapping.

The architecture also supports attribute based and region based addressing. In the case of an SAN, the physical position of nodes can be assumed to be known to the SM which maintains a table that describes a mapping between the Bluetooth Device address and its physical location in the network. The master can then address a particular region by using the Bluetooth Device address → physical location → network address lookup table. The master first identifies all such nodes and can then address them directly by figuring out their network address and putting it in the header of the Query packet. The attribute based addressing scheme shall be illustrated in the next section. The Response packet that is constructed related to any of the queries follows the reverse path. Since there is only a single path to the SM, no routing is required on the inverse path. The header however still retains the ‘hop count’ field, which is incremented at every hop on the way back, and the local address in the piconet. This allows the SM to exactly isolate from where the packet has come in the network.

4.3 Data Dissemination

Another important issue in routing, particularly in sensor networks, is that of data dissemination. Sensor networks are mostly data-centric in which the interest dissemination is performed to assign the sensing tasks to sensor nodes. There are two approaches used for this; either the ‘sinks’ broadcast the interest [14], or the ‘sources’ broadcast an advertisement for the available data and wait for a request from the interested sinks [18]. The data aggregation can be performed by using either of the following depending upon the type of application:

- i. location or region based
- ii. attribute based

The former has been discussed in the previous section. The latter can be realized by having the SM broadcast the task descriptors that are named by assigning attribute-value pairs that describe the task. As the interest is propagated throughout the sensor network, each node checks if it has any data corresponding to it. If there is, then it constructs a Response packet with the data and the associated time stamp information is included. Besides this, while the requested data is on its way back, a certain amount of in-network processing can also be done.

4.4 Healing

Another important property is that of self-healing so that a network becomes tolerant to any unexpected node or link failures. It is thus important for maintaining the required network connectivity. We propose the following approach. The SM resorts to offline processing for

calculating alternate topology configurations assuming the failure of every node/link in the SAN, one by one. We only consider single node failures here. The topology as a result of simultaneous failure of nodes must be computed online real time as the resulting possible configurations are exponential in number and hence impossible to predict, compute and store ahead of time. The information can be used to instantaneously reconstruct the topology in case of unexpected breakdowns. To detect these node failures or link losses, specific Bluetooth primitives are already defined in [1] which will be used. The ‘Link_Supervision_Timeout’ value is used by the Master or Slave Bluetooth device to monitor link loss. If, for any reason, no baseband packets are received for any particular Connection Handle for a duration longer than the set timeout value then the link is disconnected causing a ‘Disconnection_Complete_Event’ with an error code of 0x08. The moment a particular sensor node at any level detects the presence of link loss or node failure, it tries to page its slave node again. If it is able to reconnect to the node within a predetermined number of ‘Page_Timeouts’ then it implies that it was a link loss due to some transient condition (e.g., a passing vehicle on a bridge). If not, then it is assumed to be a node failure and the information is communicated to the SM, which then takes appropriate action. The SM first figures out the level in the hierarchy where the link loss or node failure has occurred. It then dismantles all the nodes at that level and reconstructs the hierarchy from that point onwards. The process is fast enough because the SM already has the required topology calculated and the clock values of each sensor node are known.

If a new node randomly joins the SAN, it will be discovered by several nodes that are executing the Bluetooth defined ‘HCI_Periodic_Inquiry’ primitive in their idle time (if they are configured to do so). The information is then relayed to the SM which then decides which of the neighboring nodes should be allowed to act as the master of the new node based on the each node’s current slave density, its buffer capacity and the link strength of the link with the new node. If the addition causes a major load imbalance, determined by the variance of the net traffic converging at the node then a new topology will have to be calculated.

4.5 Timing Accuracy

In several situations time stamping of events is required with a high degree of accuracy. One example of this could be position estimation using Kalman filters. Another example, which might have a similar requirement, could be the joint analysis of video and some other sensor data such as the strain sensor. The images captured through the camera are sometimes required to be correlated with the strain measurements to

ascertain the effect of certain kind of traffic over a bridge. As soon as the sensor node detects an event for which it is either requested or programmed for, it timestamps it with the clock value of its native clock. To construct a complete picture, data associated with any such event shall be registered with several nodes. Data from all these nodes traverse upwards with the ultimate objective to reach the data logger for further analysis or processing. To this effect, the SM requires to correlate the sensor data based on a common reference point. One can safely consider the SM's native clock for this purpose. Also, the Bluetooth specifications [1] allow the clock offset between any pair of master-slave nodes to be retrieved from the baseband using *Read_Clock_Offset* primitive. Thus as the information propagates upwards, the time stamp gets updated at every hop by using the clock offset information. The specifications allow a positive or negative clock drift of ' ρ ' (20ppm). The amount of error that accumulates depends on the frequency of 'clock offset information' updates. If we make this update a part of the scheduling process then the maximum amount by which the clocks can drift relative to each other in any two adjacent hierarchy level would be ' $2T_s \cdot 2\rho$ '. The aggregate error, by the time the packet reaches the SM, would then be ' $2T_s \cdot 2\rho \cdot h$ ', where ' h ' is the level from which the packet had originated. This quantity is of the order of microseconds. Also the clock drifts are calculated pair wise and the accumulated drift doesn't affect the accuracy of the results as long as all the inputs are compared against a common reference clock, that of the SM, and the baseband implementation of individual nodes can keep track of the clock offset information of the device with which it is interacting. Hence one can achieve a sufficient level of accuracy (microsecond level) using the proposed topology.

Besides time stamp accuracy, the amount of delay any packet suffers in the topology is also very important. In our model, this figure is directly proportional to the level at which the packet was generated. Assuming that the packet was generated at level ' h ', the amount of time it takes to reach the SM is simply twice the scheduling period times the distance traveled, i.e., ' $2T_s h$ '.

4.6 Energy Model

Low power consumption is the most stringent requirement on a sensor node. We analyze, in this section, how the power consumed by a node varies with the scheduling period of the alternate layers of the proposed network hierarchy. The model used for determining the average power consumption of the radio has been adapted from [16] and is given by the following equation:

$$P_C = N_T [P_T(T_{ON} + T_{ST}) + P_{OUT}(T_{ON})] + N_R [P_R(R_{ON} + R_{ST})] \quad (10)$$

where $N_{T/R}$ is the number of times transmitter/receiver is switched on per unit time and is equal to $(2T_s)^{-1}$ (refer to section 4.1); P_{OUT} , the output power of the transmitter; $(T/R)_{ON}$, the transmitter and receiver on time; $(T/R)_{ST}$, the transmitter and receiver start-up time and $P_{T/R}$ is the power consumed by transmitter/receiver. A detailed analysis has been presented in the next section along with the results and observations.

5 SIMULATION RESULTS

We carried out simulations to verify our proposed scheme. There are two parts to the simulation. In the first part we compare the two algorithms proposed in Section III for their effectiveness in the topology formation. The metrics of comparison that we used are the degree of load balance, denoted by *balance index*, the slave density achieved by each one of them and the number of iterations taken to converge to an optimum solution. The second part investigates the relationship between the power expended in the piconet vs the periodicity with which the scheduling is performed between two adjacent hierarchy levels in the topology. We then also discuss an optimal point of operation based on buffer considerations.

To evaluate the effectiveness of our solution, the entire set of simulations was carried out on a set of rectangular sensor grids – modeled to span a portion of a bridge. The three grids (one for temperature, one for pressure and one for vibration) were placed together with a slight offset between them so that the nodes are uniformly spread across the area. To model the link strength calculation, we assumed a linear relationship with the distance, but the same can be extended to any functional relationship without the loss of generality as long as the link strength decreases with the distance and vice-versa. We also assume that $\alpha = \beta = \gamma = \delta = 1$ to have equal contribution from different components. The simulations were carried out with the following values assigned to each of the parameters. We did not do any sensitivity analysis of the proposed values for the different parameters. We tried different configurations and in all the cases the same solution was reached as with the values highlighted here. The only difference was in the time that it took to converge.

- i. The total number of nodes participating in the network was varied from 100 to 1000 with a step size of 100 with portion of bridge on which they were placed to be of dimensions 100m x 20m.

- ii. The load factors for the temperature, pressure and vibration sensor grid were assumed to uniformly spread across the interval 1-100, 101-200 and 201-300 bits per sec, respectively.
- iii. The communication range of a node was assumed to be 20m, which implies that the link strength goes to zero at that distance.
- iv. All the nodes were assumed to have an identical buffer capacity of that of 4KB.
- v. For the Simulated Annealing algorithm, the temperature was varied from 5000 to 1 with a scaling factor of 0.5. This choice resulted from several test runs that showed similar results with faster convergence for this value.
- vi. For computing the energy expended by the entire piconet the traffic incident and leaving each of the nodes in the final topology was taken into consideration. Based on this and the maximum asymmetric data rate of 721 Kb/s using DH5 packet, the ‘On’ time of both the transmitter and receiver was calculated. The values given to P_T , P_{OUT} , P_R , T_{ST} , R_{ST} were 90mW, 1mW, 180mW, 100us and 100us respectively.
- vii. The load balancing at any node is measured by taking the variance (balance index) of the aggregate bandwidth at each of the links that are incident on the node.
- viii. The slave density, mentioned before, was calculated using the expression

$$\rho_s = \frac{1}{n_p} \sum_{k=1}^K (n_{pk} \times l) \quad (11)$$

where n_{pk} refers to the number of piconets

having k slaves and n_p is given by $n_p = \sum_{k=1}^K n_{pk}$.

Also, it was assumed that all the nodes were present at the time of execution of the algorithm.

Table I shows the link load distribution at the top of the tree hierarchy (level 1) from link one through link seven ($k=7$) achieved through SA. From the standard deviation of the link bandwidths we can see that the load is reasonably well distributed over the 7 links for most of the scenarios (100 to 1000 nodes).

Fig. 2 gives a comparison between the load-balance achieved by using the two algorithms proposed in section 3. The load balance is estimated using a metric called ‘balance index’ calculated using the following expression

$$\text{B.I.} = (1 + \sigma_d)^{-1} \quad (12)$$

where ‘ σ_d ’ is the standard deviation of the link bandwidths converging into the node in consideration. As evident from the figure, SA is able to do a much better job because of its ability to jump out from local minima on an n -dimensional terrain. The GS algorithm searches the terrain quite aggressively and has a higher probability of getting stuck in a local minimum. Also, the GS does not take into consideration the non-linearity introduced by the selection criterion of the elements which favors the elements with ‘ k ’ highest SAFs. The energy of a configuration not only depends on the movement of one particular element but also the relative placement of the remaining elements. The GS on the contrary involves the movement of elements in a linear fashion and then, only one at a time. (For both, Fig. 2 and Fig. 3, the regression lines for SA and GS show the average case behavior considered over a number of simulations with different initial seeds. The curves plotted refer to just one such simulation.)

Fig. 3 compares the slave density between SA and GS, which is directly dependent on the number of piconets in the final topology. We can observe that as the number of nodes increases, the slave density of SA starts to catch up with that of GS and the asymptotic behavior of SA is slightly better than GS.

Fig. 4 shows the trend of power expended within the sensor network in terms of the periodicity with which the alternate levels of the topology are scheduled. As the period increases the net power consumed comes down. It thus presents a direct trade-off between the delay requirements (section 4.5) and the energy drainage. The reason for this behavior is that as the frequency of switching increases, the power consumed by the radio to switch on and switch off, increases while that of the data transmission remains the same. The heavy line (legend Optimal) shows a sequence of the ideal points of operation for different node densities. The criterion of selecting those values comes from the buffer limitation at each of the nodes. Since all the nodes are assumed to have equal buffer capacities, hence to avoid buffer flows at any of the nodes in the topology, we consider the nodes at level 1 which carry the highest amount of traffic (eq. (9)). The limit is based on the ‘time’ it takes for the buffer to overflow which is equal to twice the scheduling period. The exercise was performed for topologies with different node densities and the profile was obtained which also gives a minimum bound on the power consumption in a topology for a given buffer capacity. (It should be noted that interference among different piconets has not been taken into account in this analysis and remains to be analyzed in detail in the future work.)

6 CONCLUSIONS

The proposed solution takes into account several factors that play a key role in the deployment and performance of a SAN. The topology discussed above represents a logical organization of the sensor nodes and also provides a framework for managing sensor networks. We also observed the effects of the scheduling period on the overall performance in terms of energy and buffer usage. Future work will focus on analyzing the effects of inter-piconet interference on the rest of the system metrics such as energy consumption, throughput and delay. We also plan to investigate the deployment of multiple sensor hubs and forming a tree of smaller scatternets with similar criteria being used to form the tree but at a macro level. The mobility of the Super Master can also be considered which would allow the collection of data from a larger geographic region with the same number of resources.

The deployment of sensor networks based on Bluetooth technology will mostly depend upon the implementation and support of scatternets by commercial vendors. In the current scenario, there is a considerable lack of effort in this direction as the industry is concentrating more to capitalize on the existing market segments which limit themselves to either point to point connection or at most the piconet functionality serving different profiles such as a PAN and wireless Headsets.

REFERENCES

- [1] "Bluetooth specifications version 1.1," <http://www.bluetooth.com/dev/specifications.asp>
- [2] "Development of Wireless Sensor Technology for Machine Monitoring," http://ieee1451.nist.gov/Workshop_04June01/Bocko.pdf.
- [3] Theodoros Salonidis, Pravin Bhagwat, Leandros Tassioulas, and Richard LaMaire, "Distributed topology construction of Bluetooth personal area networks," *Proceedings of IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.
- [4] Ching La, and Kai-Yeung Siu, "A Bluetooth Scatternet Formation Algorithm," *Proceedings of IEEE Symposium on Ad Hoc Wireless Networks 2001*, San Antonio, Texas, November 2001.
- [5] Godfrey Tan, Allen Miu, John Guttag and Hari Balakrishnan, "Forming Scatternets from Bluetooth Personal Area Networks," *MIT Technical Report*, MIT-LCS-TR-826, October 2001.
- [6] Gergely V. Zaruba, Stefano Basagni, and Imrich Chalantac, "Bluetrees - Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks," *Proceedings of IEEE International Conference on Communications (ICC'01)*, pp. 273-277.
- [7] Lakshmi Ramachandran, Manika Kapoor, Abhinanda Sarkar, Alok Aggarwal, "Clustering Algorithms for Wireless Ad Hoc Networks," *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Boston, MA, August 2000, pp 54-63.
- [8] Mario Gerla, Rohit Kapoor, Manthos Kazantzidis, Per Johansson, "Ad Hoc Networking with Bluetooth," *Proceedings of WMI at ACM Mobicom 2001*, Rome, Italy, July 2001.
- [9] Kirkpatrick, S., Gelatt, C.D., and Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 671-680.
- [10] V. Rodoplu, T.H. Meng, "Minimum energy mobile wireless networks," *IEEE Journal of Selected Areas in Communications*, 17 (8) (1999), pp. 1333-1344.
- [11] L. Li, J.Y. Halpern, "Minimum-energy mobile wireless networks revisited," *Proceedings of IEEE International Conference on Communications (ICC'01)*, Helsinki, Finland, June 2001.
- [12] K. Sohrabi, J. Gao, V. Ailawadhi, G.J. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications*, October 2000, pp 16-27.
- [13] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-efficient communication protocol for wireless sensor networks," *Proceedings of the IEEE Hawaii International Conference on System Sciences*, January 2000, pp. 1 - 10.
- [14] C. Intanagonwivat, R. Govindan, D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Proceedings of ACM MobiCom '00*, Boston, MA, 2000, pp. 56-67.
- [15] A. Woo, D. Culler, "A transmission control scheme for media access in sensor networks," *Proceedings of ACM MobiCom '01*, Rome, Italy, July 2001, pp. 221-235.
- [16] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," *Proceedings of ACM MobiCom '01*, Rome, Italy, July 2001, pp. 272-286.
- [17] S. Hedetniemi, A. Listman, "A survey of gossiping and broadcasting in communication networks," *IEEE Networks*, 18 (4) (1988) 319-349.
- [18] W.R. Heinzelman, J. Kulik, H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," *Proceedings of ACM MobiCom '99*, Seattle, Washington, 1999, pp. 174-185.
- [19] <http://today.cs.berkeley.edu/tos/>
- [20] <http://www.ieee802.org/15/pub/TG4.html>

Table I. Link Bandwidth Distribution Across the Links converging at the root

Link Nodes	L-1 Kbps	L-2 Kbps	L-3 Kbps	L-4 Kbps	L-5 Kbps	L-6 Kbps	L-7 Kbps	Std dev
100	2.07	2.59	2.10	2.13	1.91	2.12	1.56	0.305
200	4.59	4.35	4.90	3.06	4.19	2.66	4.08	0.713
300	6.09	6.77	4.86	7.75	5.95	6.73	6.00	0.896
400	7.48	9.08	9.12	9.40	7.99	7.93	8.13	0.737
500	10.56	9.91	9.55	13.33	12.11	8.36	10.28	1.654
600	12.02	14.62	11.94	12.21	11.94	12.78	14.31	1.155
700	13.28	18.34	14.11	14.72	12.90	15.34	15.14	1.794
800	14.21	19.24	14.65	20.38	17.22	16.65	16.66	2.238
900	16.92	19.92	17.77	21.38	22.28	18.72	17.86	2.000
1000	20.33	27.36	19.94	19.88	25.09	21.54	15.96	3.751

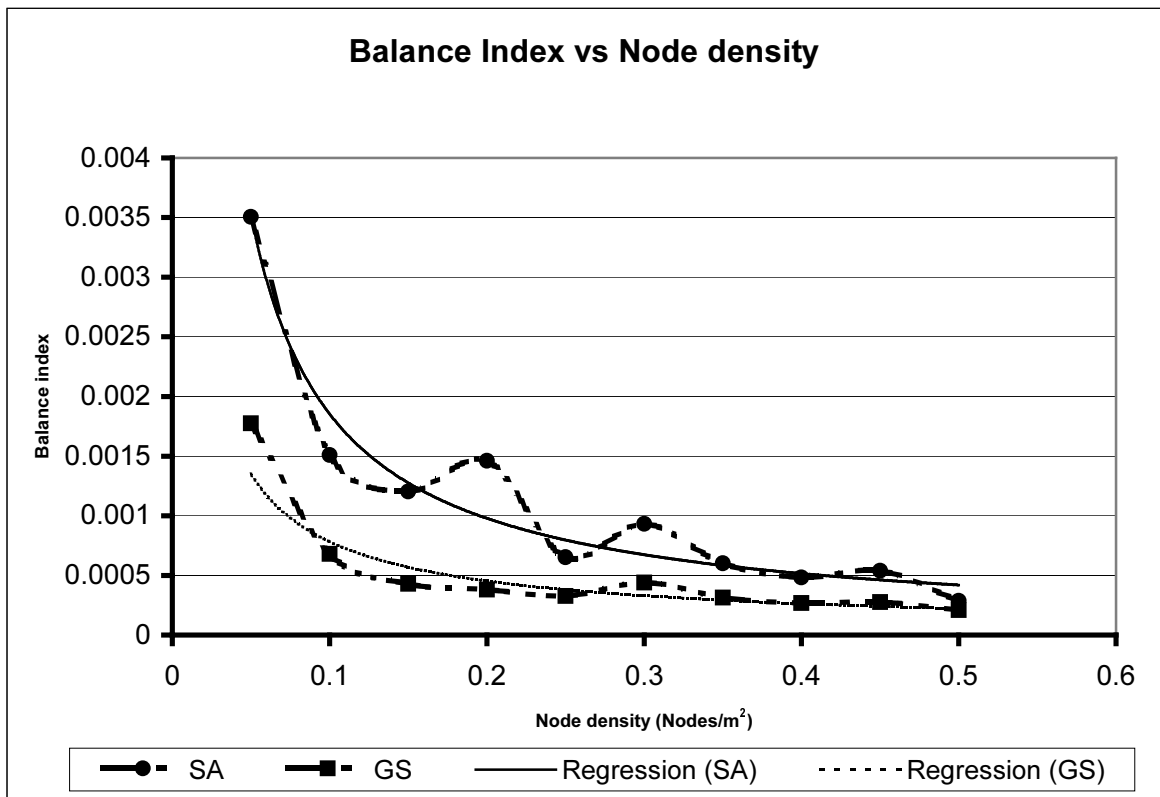


Figure 2. Balance Index vs Node Density

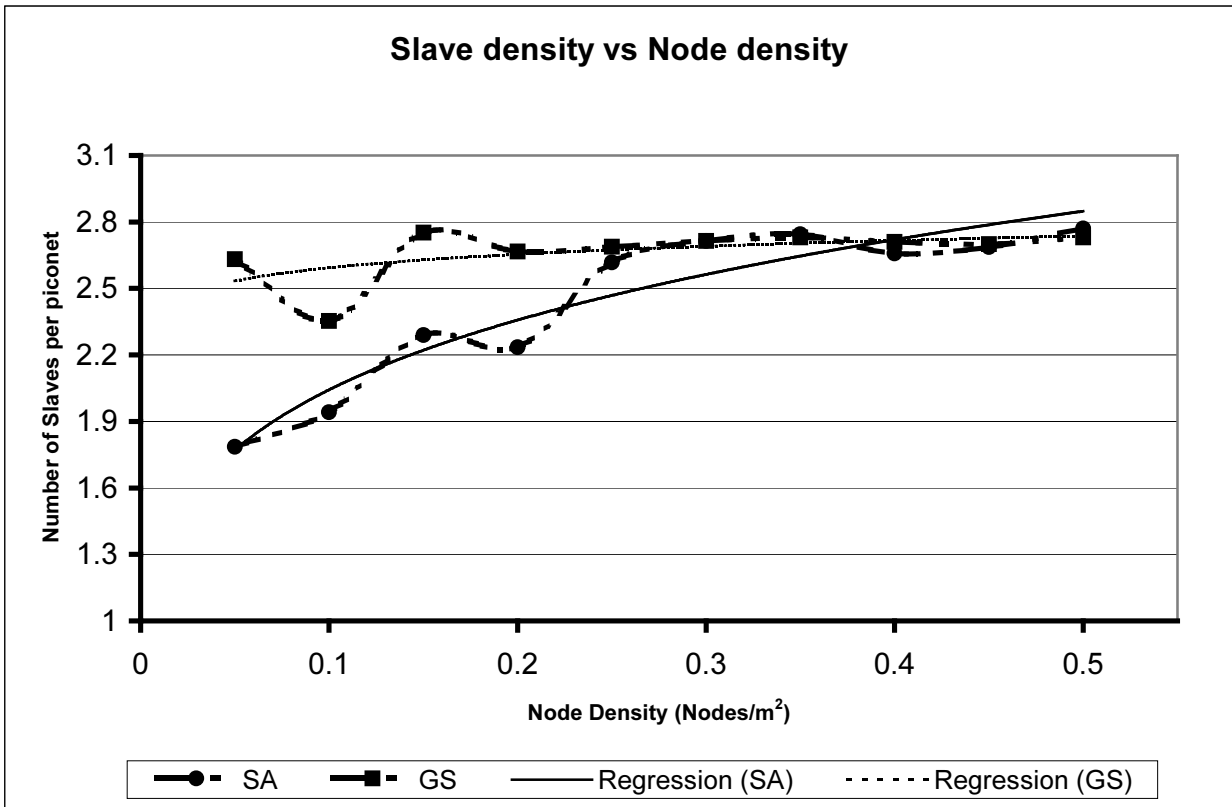


Figure 3. Variation of Slave Density with Number of Nodes

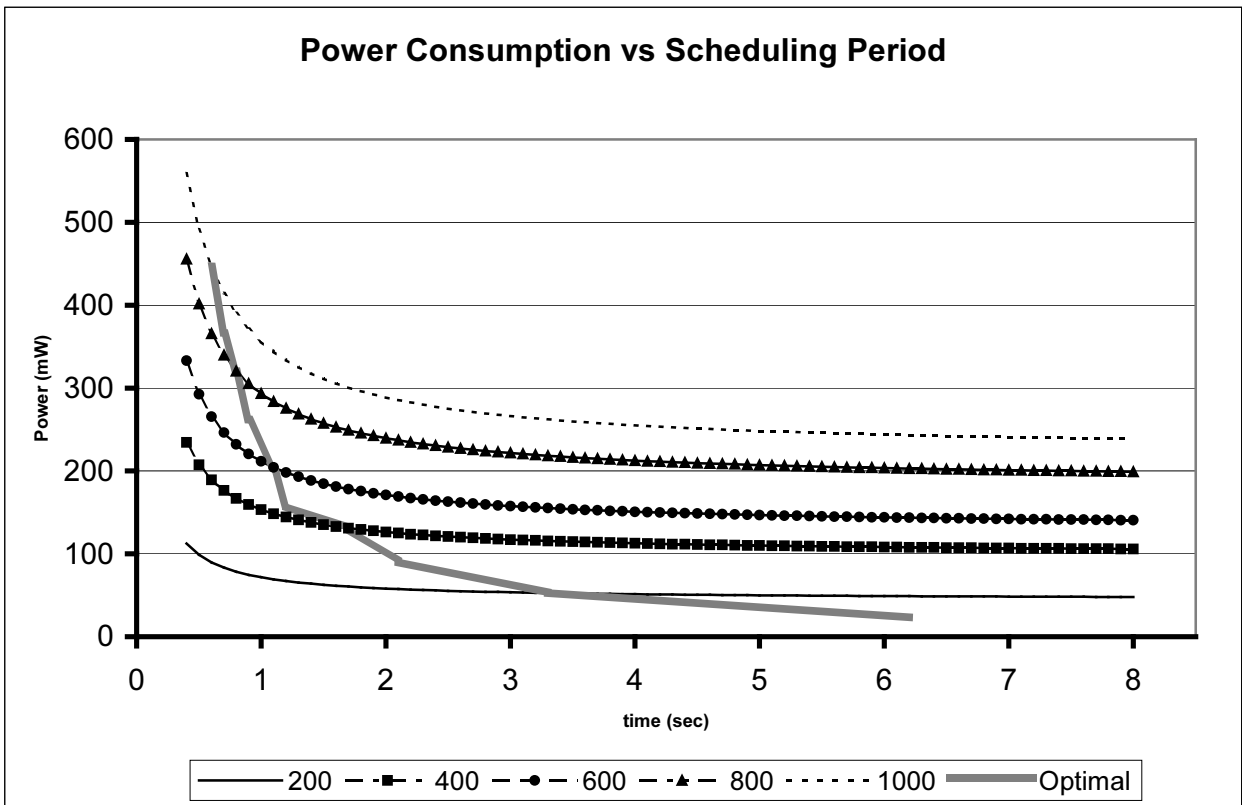


Figure 4. Variation of Power Consumption with Scheduling Period