CS-236

Project Proposal

Anirban Banerjee
UCR ID#860750803
Anirban@cs.ucr.edu

SenSeRT: A Sensor Network SRT Initialization Protocol

SenSeRT: A Sensor Network SRT Initialization Protocol

Abstract:

This project proposal outlines the various parent selection strategies which may possibly be employed for creation of a protocol specifically helpful in constructing *Semantic Routing Trees* in a Sensor Network environments. We evaluate five different strategies and in terms of power consumption and maximum and average path lengths for query probes in the network. We show that our hybrid8:2 strategy performs quite better than other strategies with a 94% correlation, and is close to an optimal attribute based Semantic Routing Tree topology. Semantic Routing Tree, essential for management of queries dealing with range-based attributes are an integral part of sensor network architectures. In the absence of this routing structure it is infeasible, with regards to energy depletion due to redundant transmissions, for nodes to process range based queries over particular attributes. Nodes participating actively in a sensor network need to understand clearly whether they need to forward the range based queries to lower levels of the sensor hierarchy. Semantic Routing Trees is but a much-behooved panacea in this regard. The role of these routing structures has been acknowledged, as overlays, and has been implemented on pilot sensor implementations. Unfortunately, the underpinnings for a Semantic Routing Tree initialization have not been specified clearly, as of today. This project attempts to underline this very vital need to develop an unambiguous Semantic Routing Tree initialization protocol and presents SenSeRT in this regard.

1. Introduction

In the past few years, smart-sensor devices have matured to the point that it is now feasible to deploy large, distributed networks of such sensors [1], [2], [3], [4]. Sensor networks are differentiated from other wireless, battery powered environments in that they consist of tens or hundreds of autonomous nodes that operate without human interaction (e.g. configuration of network routes, recharging of batteries, or tuning of parameters) for weeks or months at a time. Furthermore, sensor networks are often embedded into some (possibly remote) physical environment from which they must monitor and collect data. The long term, low power nature of sensor networks, coupled with their proximity to physical phenomena lead to a significantly altered view of software systems than that of more traditional mobile or distributed environments.

In this project, we are concerned with *Semantic Routing Tree* (SRT) [11] initialization for efficient query processing in sensor networks. Significant amount of research has been directed towards highlighting the benefits of a query processor-like interface to sensor networks and the need for sensitivity to *limited power and computational resources* [5],

[6], [7], [8], [9]. Prior systems, however, tend to view query processing in sensor networks simply as a power constrained version of traditional query processing: given some set of data, they strive to process that data as energy-efficiently as possible. Typical strategies include minimizing expensive communication by applying aggregation and filtering operations inside the sensor network – strategies that are similar to push-down techniques from distributed query processing that emphasize moving queries to data. *We intend to build an SRT and re-use it for probing the network for queries on similar attributes. The refresh period, after which the SRT needs to be reconstructed can be tweaked according to the needs of the application environment.*

An SRT is a routing tree designed to allow each node to efficiently determine if any or all the nodes below it will need to participate in a given query over some attribute "*A"* [13]. Traditionally, in sensor networks, routing tree construction is done by having nodes pick a parent with the most reliable connection to the root. With SRTs, it is but obvious that the choice of a parent should necessarily include some semantic parameters. In general, SRTs are most applicable in situations in which there are several parents of comparable link quality. A link quality based parent selection algorithm as described in [10], can be employed in conjunction with the SRT to prefilter the set of parents made available to the SRT.

Conceptually and SRT is an index over "*A"* [12] that can be used to select nodes that have data relevant to the query. Unlike traditional indices, however the SRT is an overlay on the network. Each node stores a single unidimensional interval representing the range of "*A"* values beneath each of its children. When a query "*q"* with a predicate over "*A"* arrives at a node "*n"*, "*n"* checks to see if any child's value of "*A"* overlaps the query range of "*A"* in "*q"*. If so, it prepares to receive results and forwards the query. If no child overlaps, the query is not forwarded. Also, if the query also applies locally (whether or not it also applies to any children) "*n"* begins executing the query itself. If the query does not apply at n or at any of its children, it is simply forgotten.

A simplistic means of building an SRT [11], can be conceptualized as a two phase process: first the *SRT build request* is flooded (re-transmitted by every mote until all motes have heard the request) down the network. This request includes the name of the attribute "*A"* over which the tree should be built. As a request floods down the network, a node "*n"* may have several possible choices of parent, since, in general, many nodes in radio range may be closer to the root. If "*n"* has children, it forwards the request on to them and waits until they reply. If "*n"* has no children, it chooses a node "*p"* from available parents to be its parent, and then reports the value of "*A"* to "*p"* in a *parent selection message*. If "*n" does* have children, it records the value of "*A"* along with the child's ID. When it has heard from all of its children, it chooses a parent and sends a selection message indicating the range of values of "*A"* which it and its descendents cover. The parent records this interval with the ID of the child node and proceeds to choose its own parent in the same manner, until the root has heard from all of its children. Figure 1 shows an SRT over the latitude. The query arrives at the root, is forwarded down the tree, and then only the dark nodes are required to participate in the query (note that

node 3 must forward results for node 4, despite the fact that its own location precludes it from participation.)

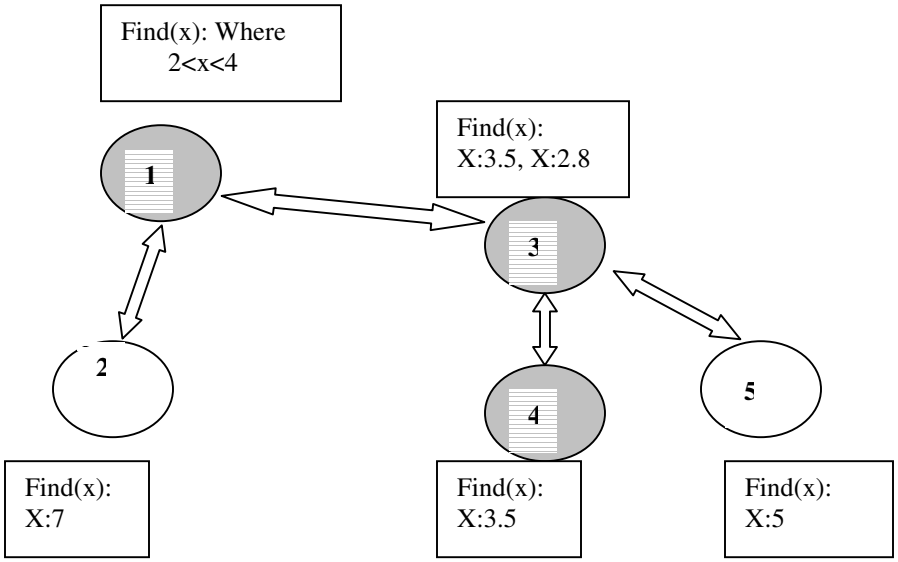

Figure 1. : A SRT being used to direct queries to only those nodes which have data relevant to the attribute X, only the dark nodes participate.

2. Maintaining SRTs

Even though SRTs are limited to constant attributes, some SRT maintenance must occur. In particular, new nodes can appear, link qualities can change, and existing nodes can fail. Node appearance and link quality change can both require a node to switch parents. To do this, it sends a parent selection message to its new parent, n. If this message changes the range of "*n*'s interval, it notifies its parent; in this way, updates can propagate to the root of the tree. To handle the disappearance of a child node, parents associate an *active query id* and *last epoch* with every child in the SRT (recall that an epoch is the period of time between successive samples.) When a parent "*p"* forwards a query "*q"* to a child "*c"*, it sets *c*'s active query ID to the ID of "*q"* and sets its last epoch entry to 0. Every time "*p"* forwards or aggregates a result for "*q"* from "*c"*, it updates *c*'s last epoch with the epoch on which the result was received. If "*p"* does not hear "*c"* for some number of epochs t, it assumes "*c"* has moved away, and removes its SRT entry. Then, "*p"* sends a request asking its remaining children retransmit their ranges. It uses this information to construct a new interval. If this new interval differs in size from the previous interval, "*p"* sends a parent selection message up the routing tree to reflect this change. Finally, we note that, by using these maintenance rules proposed, it is possible to

support SRTs over non-constant attributes, although if those attributes change quickly, the cost of propagating changes in child intervals could be prohibitive.

2.1 Evaluation of Benefit of SRTs

The benefit that an SRT provides is dependent on the quality of the clustering of children beneath parents. If the descendents of some node n are clustered around the value of the index attribute at n, then a query that applies to n will likely also apply to its descendents. This can be expected for geographic attributes, for example, since network topology is correlated with geography. There are predominantly three policies for SRT parent selection [11]. In the first, *random* approach, each node picks a random parent from the nodes with which it can communication reliably. In the second, *closest-parent* approach, each parent reports the value of its index attribute with the SRT-build request, and children pick the parent whose attribute value is closest to their own. In the *clustered* approach, nodes select a parent as in the closest-parent approach, except, if a node hears a sibling node send a parent selection message, it *snoops* on the message to determine its siblings parent and value. It then picks its own parent (which could be the same as one of its siblings) to minimize spread of attribute values underneath all of its available parents.

2.2 SRT Summary

SRTs provide an efficient mechanism for disseminating queries and collecting query results for queries over constant attributes [11]. For attributes that are highly correlated amongst neighbors in the routing tree (e.g. location), SRTs can reduce the number of nodes that must disseminate queries and forward the continuous stream of results from children by nearly an order of magnitude.

3. SPP : An Overview

[17] Sequence Packet Protocol (SPP) is the primary transport-layer protocol in the Xerox Network Systems (XNS) and is employed by the ChipCon CC 1010. It provides reliable, flow-controlled, two-way transmission of data for an application program. It is a byte-stream protocol used to support the **SOCK_STREAM** abstraction. The SPP protocol uses the standard Network Systems (NS) address formats. The SPP layer presents a byte-stream interface to an application or user process. As a byte-stream protocol, SPP is used to support the **SOCK_STREAM** mechanism for interprocess communication. Sockets using the SPP protocol are either active or passive. By default, SPP sockets are created active. The following conventions apply to using active and passive sockets:

- Active sockets initiate connection to passive sockets.
- Only active sockets may use the **connect** subroutine to initiate connections.
- To create a passive socket, an application must use the **listen** subroutine after binding the socket with the **bind** subroutine.
- Only passive sockets may use the **accept** subroutine to accept incoming connections.

The following table illustrates the table parameters available for SPP in general.

Source connection ID	Reference number used to identify the source end of a transport connection. This protocol establishes Connection IDs at connect time to distinguish between multiple transport connections.
Destination connection ID	Reference number used to identify the target end of a transport connection.
Sequence number	Sequence number of the packet. Each successive packet transmitted and acknowledged on the transport connection must have a sequence number one higher than the previous sequence number.
Acknowledge number	Sequence number of the last packet that the protocol received properly. Each side of the transport connection uses its own sequence of numbers for transmitted packets, resulting in sequence and acknowledge numbers in the same packet generally being out of phase with each other.
Credit	Number of unacknowledged packets that the other side of the transport connection can send.
Transport control flag	When set (value of 1), the packet is used for transport control.
Acknowledge required flag	When set (value of 1), an immediate acknowledgement is requested.
Attention flag	When set (value of 1), the packet is sent regardless of the credit advertised by the destination.
EOM	End of message flag. When set (value of 1), a logical end of message stream is denoted.
Datastream type	Reserved field ignored by the SPP transport layer. SPP provides the datastream type for use by higher level protocols as control information.

Details on RF networking implemented in the CC1010 software library [18], follow.

1. The SPP (Simple Packet Protocol) functions/macroes in CUL implements the basic RF link protocol, not network/client arbitration. The idea is to implement such arbitration 'above' the CUL level.
2. Chipcon has not yet implemented a full RF network arbitration level in the CC1010 libraries. However, the Chat example should give some indication on how this can be done in a very simple manner:

   1. [A] sends a SPP packet to [B], using sppSend().
      [B] receives it, using sppRecv().

   2. [B] is then required to send an acknowledge packet back to [A].
      In this [B]->[A] packet [B] must then include a channel access request
      (if it wants to send another packet to A).
      [A] is then required to wait a certain time for [B] to send its packet.

Collision detection is mainly implemented by ACK timeout limits, packet sequence tracking, etc, while network arbitration is done by unique addressing of each client in the network. This is assigned by the RF network server (which is selected during set-up/initialisation). The efficacy of this protocol is highlighted by [19].

4. The SenSeRT Protocol

In this section we go on to describe our view of an SRT initialization protocol [14], [15], [16]. Even though SRTs have been the focus of research, a deep understanding of how exactly a sensor network can evolve an SRT remains elusive. We attempt to lift the veil from the proverbial “implementation issue” in this project. The roadmap for this project is detailed in the following paragraph.

We list the steps to develop the SenSeRT protocol, highlighting at each step the important issues and how we will proceed to solve them.

1) *Assumption*: All sensors sprinkled over an area are in radio range of each other, this assumption is certainly not a “profanity” in terms of sensor capabilities as may seem at a first look. The ChipCon 1010 sensor on which the RISE project revolves aims to use MMICs to increase the radio transmission range of each node in the next phase. Also all nodes except the initiator are in listening mode. All Nodes listen to a particular pre-agreed transmission frequency. Each node has the capability to change its transmission or Receiving frequency on the fly. The number of such frequencies is limited to three. The RISE platform built on the ChipCon 1010, can operate at 868/915 MHz or 2.4 GHz, this modeling is in convergence with the actual sensor part of the RISE platform.
2) *Who starts the communication*: This is fundamental to formation of a structured routing tree. Most sensor implementations assume different types of sensors with varied capabilities in terms of energy availability and computational capabilities. We do not agree with this approach. Such a design is flawed for a number of basic reasons. Commercially speaking, every major chip manufacturer, or SoC provider

spends millions on developing their systems and to customize their nodes for deployment at different "hierarchical levels" seems silly, and financially unwise. We thereby assume the presence of only one special initiator node. All other nodes have the same capabilities. This node is responsible for starting the SRT formation procedure; no other node can begin this process.

3) *How do we form the hierarchy*: This is done in a lock-step fashion. The initiator node floods an SRT-Make packet with TTL 1, and its ADMIN-ID of 1, to its one-hop neighbors to let them know that it wants to initiate the SRT creation process. Any node that hears this SRT-Make, understands that it now has to take on the baton for the SRT creation. These one hop neighbors retransmit the SRT-Make packet with TTL 1, and a COMMON-ID 2, specifying that the request came from a generic pool of nodes and not the master controller. The rationale for using a TTL of 1 in this stage stems from the fact that a *shorter transmission radius ensures lesser chances of transmission collisions.* Any node which receives such a request will wait till a random backoff timer set between, 0.1 sec to 1 sec before replying to the request with a MY-RANGE message. This backoff timer is essential for the node to further the SRT-Make packet forward to the other nodes and at the same time listen to requests for SRT-Makes from its "temporal" parents. Each secondary node makes a decision of choosing parents at this stage, the major highlight lies in choosing not only a primary parent but also a secondary parent. With the primary parent the node will communicate on a particular frequency, having the best link quality indicator and with the secondary parent it will maintain a link on a different frequency, obviously accepting a lesser degree of link quality with it. The node will forward its MY-RANGE values to both its parents, so that fault tolerance is built into the system in case the primary parent "fails". Of course the secondary parent will only append the MY-RANGE value from this node after it receives an explicit HOOKUP message from it. This concept is essential for synchronized data fusion at each "temporal" parent, in case nodes forwarding their ranges to it, miss out on their timers and fail to forward their ranges in time for the parent to compile and summarize the information and hand it on to a different node.
4) *When are the Attribute ranges specified*: The attribute ranges are specified at the very onset, during flooding of the SRT-Make packet itself. In fact the SRT-Make packet is a probe, query, by itself. Consider for instance, a sensor network deployed for measuring temperature. The initial SRT-Make packet would actually ask each sensor to report back its range of currently observed temperature.
5) *How are the Ranges exchanged*: The ranges are exchanged in response to the SRT-Make packet using the MY-RANGE response packet.

Figure 2 illustrates the SenSeRT protocol. Figure 3, provides a staggered temporal view of the protocol. The full protocol specification for this initialization mechanism will be the core idea of this project.

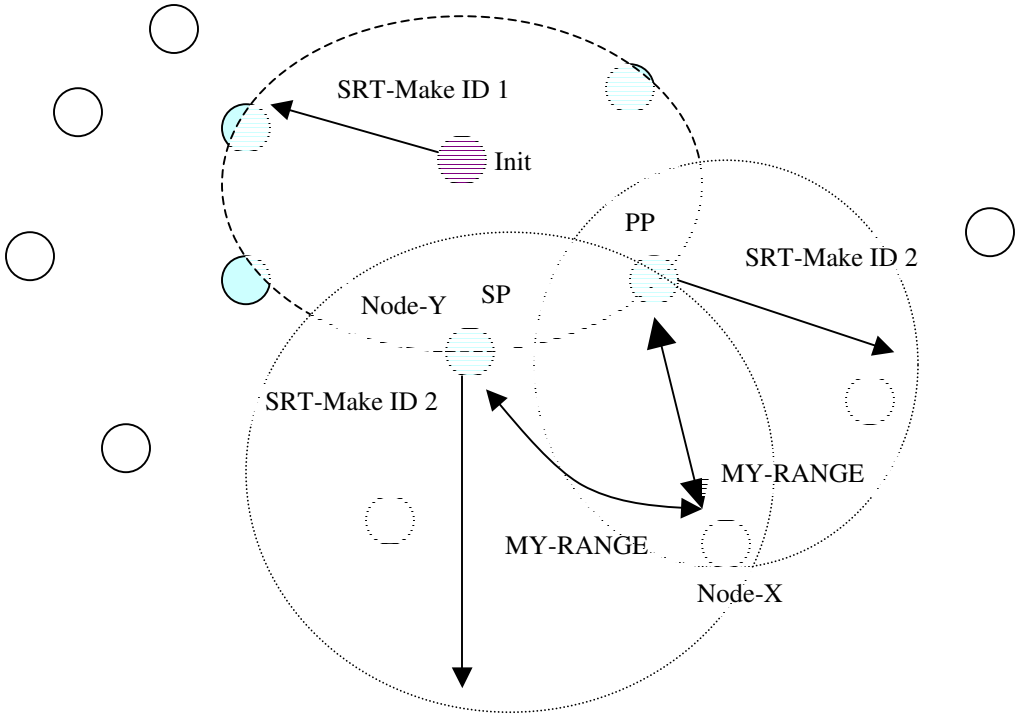

Figure 2. : The SenSeRT protocol in action. The Init (Initiator) node, intitaiates the SRT-Make procedure by flooding its 1 hop neighbors with an <SRT-Make ID 1>, packet. The 1 hop neighbors receive this message and relay this same signal, albeit with their own COMMON ID, 2 in this case. Node-X which receives requests for it range from two temporal parents, waits for a buffer time before responding so that it may gauge the link quality of nodes requesting it for its range value. Once it chooses the primary parent (PP) it links with it by sending over its range values, and the aggregated ranges of other nodes that have chosen it as its parent and have passed their data to it. It however links up to the secondary parent (SP) via a different frequency and passes on its range to it to, in case PP fails SP can link up with Node-X after receiving a HOOKUP packet from Node-X.

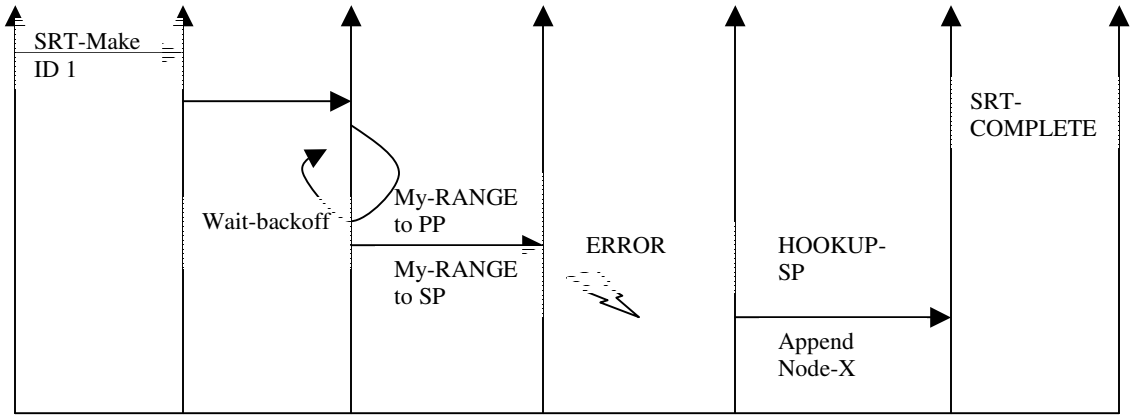

Figure 3. : The SenSeRT protocol; staggered protocol operation.

4. Mathematical Formalization

Problem: *How can we best build an SRT (Semantic Routing Tree) in a sensor network.* We want to construct an SRT and use it multiple times, how is the parent selection strategy that we employ going to effect the sensor network.

Formulation: All nodes in the sensor network are powered up and are listening for radio signals, the SRT_MAKE request. The initiator node injects a query into the system, "Generate temp reading", this trickles down the complete network creating the layered structure. The lowest level nodes reply to the SRT_MAKE request and relay their temperature readings to their selected parents. We assume that there are no time-outs at each node leading to missing out on the information sent by a lower layer node to its parent while creating the SRT. We do not address data fusion issues, how will each node know that all its children have replied and that it should now send its range value to "its" parent. We simply consider the effect of parent selection strategies on the SRT we try to build. Each node stores the values of "attribute" or range of attribute values that its children correspond to, and its own reading is integrated into this range.

There is a special initiator node with $L_i=0$, (node level ID).

L_j= { 0 iff node is initiator for SRT_CREATE

Else = $L_i + 1$, where $L_i = \min \{\forall L_i$ where $i,j \in N$ and $R_i >= W(i,j)\}$

R_i be the range of transmission of radio signal for node i.
W(i,j) be the weight (radial distance) of node i from node j.

{This allows for a layered ring structure via which we can classify nodes in levels 1 to 4 (Say)}

We can explore some parent selection strategies that will effect on how the SRT is formed.

1) Random parent selection

$P_i = j$ iff $R_j > W(i,j)$ and $L_j <= L_i$ { P_i , parent of node i}

2) Best Link based selection

$P_i = j$ iff $R_j > W(i,j)$ and $L_j <= L_i$ and $F_j = \max\{\forall F_p$, where $p \in N$ and $R_p >= W(i,p)\}$

{F_p is the fitness of node p}

3) Best attribute value-based selection

$P_i = j$ iff $R_j > W(i,j)$ and $L_j <= L_i$ and $F_j = \min\{\forall A_p$, where $i,p \in N$, $R_p >= W(i,p)$, we compute $| A_p - A_i |\}$

$\{A_p$ is the Attribute value of node p$\}$

4) Hybrid1:1

$P_i = j$ iff $R_j > W(i,j)$ and $L_j <= L_i$ and $F_j = \max\{\forall A_p$, where $i,p \in N$, $R_p >= W(i,p)$, we compute $(1-(| A_p - A_i |)/N_f) + (F_j)\}$

5) Hybrid8:2

$P_i = j$ iff $R_j > W(i,j)$ and $L_j <= L_i$ and $F_j = \max\{\forall A_p$, where $i,p \in N$, $R_p >= W(i,p)$, we compute $0.2(1-(| A_p - A_i |)/N_f) + 0.8(F_j)\}$

future work will concentrate on developing a window based scheme, adhering to the following formulation.

$P_i = j$ iff $R_j > W(i,j)$ and $L_j <= L_i$ and $F_j = \min\{\forall A_p$, where $i,p \in N \& R_p >= W(i,p)$, $(| A_p - A_i |)$ & $(SD_t - SD_{t-1})$ $\}$

$\{SD_p$ is the Statistical deviation in mean of the range at each node, in cycle p$\}$

Efficacy of a particular parent selection scheme is gauged by four main parameters, the maximum path length, which provides for an upper bound on the delay characteristic of the network. Path length is calculated by considering the number of radio links, modeled as edges in the graph that a query has to traverse. The Average path delay, provides a birds eye view of the network and how the strategy works as a whole. The Maximum Energy consumed by a query to travel from a level 4 node to the sink at level1, again provides for a worst case view of how soon the fringe elements in the network might start to fail due to excessive power depletion. Energy consumed by a query to reach the sink from a source is calculated in an iterative manner. The source expends Ei energy in order to process the packet and at each subsequent hop the receiver or relay node has to expend Ei as well as the energy needed to fuse the data with its own metrics. The energy expended to merge incoming data with one's own metrics have been assumed to be equal. Thus if a node at level 4 spends Ei energy to generate the reading and passes on the query to another node, the receiver has to spend Ei+(Ei), energy to relay it. Similarly the next node on the path spends Ei+(Ei+(Ei)) amount of energy. We also provide a comparison for the average energy consumption for a query packet to traverse the network from a source to a sink for all the five schemes.

5. Experimental Results

This section describes the results from simulations formulated to demonstrate the various parent selection strategies and their consequence on real-life parameters such as average and maximum power consumption in the sensor network and average and maximum path length which provides an idea of maximum delay characteristics for such networks. The simulation strategy deals with nodes at 4 levels of hierarchy, levels 1 to 4, each containing a number of sensor nodes. The first level comprises of the single initiator node, the second comprises of 4 nodes while the third and the fourth can contain anywhere from 10 to 400 nodes. Extensive simulations have been carried out by varying the number of nodes in layer 3 and layer 4 where one would naturally expect to find the maximum node density and their effects have been presented in graphical format. We evaluate five possible strategies for parent selection in SenSeRT, which are

- *Random* : Upon receiving a SRT_MAKE request the node is free to choose a random parent which can be a peer, at the same level, or a node at a higher level. However a node may not choose any other at a lower level and also may not choose one to be its parent if the the difference between their corresponding layers span more than 2 layers.
- *Bestlink* (BL) : The node can choose parent based on the RF link characteristics of the node-parent link. This strategy is optimal if the sensor network has extremely heterogeneous distribution of computing and power resources spread over a large area. This strategy would also be critical for mission critical applications which need hard performance constraints from the network.
- *BestAttribute* (BA) : The node can choose parent based on the closeness of the attribute being probed by the initial SRT_MAKE request. This strategy is optimal in the sensor network since it inherently minimizes the search area through which the query has to be flooded every time we would like to update the SRT. This strategy strategy yields the optimal performance both in terms of power consumption and delay characteristics. However for real-world applications, where link quality parameters can make the difference between being able to sense and process event data successfully and banging one's head on an RF analyzer, it is imperative that some amount of intelligence be associated ith the parent selection strategy in SenSeRT which would also favor links with better RF quality characteristics.
- *Hybrid1:1* : The node can choose a parent based on a hybrid strategy which gives equal weight to the best link characteristics of potential parents as well as to closest attribute value for the parent under consideration. A cumulative evaluation parameter is found by the linear sum of the benefits accrued due to thiese parameters and the node with the highest value is chosen as the parent.
- *Hybrid8:2* : This scheme is identical to the one described above except that it deals with the distribution of weights in a different manner. It splits the weights in an 8:2 fashion over the link quality indicator Vs the closest attribute indicator. The major motivation for this hybrid scheme is to observe if performance in terms of power or delay decreases signifiucantly if we would like to build in an amount of fault tolerance by choosing more stable links, which might not always be the optimal according to selection only based on the attribute values.

Below we demonstrate the effects of varying the number of nodes at the lowest two dense levels on the average and maximum power consumption in the network as well as the maximum path length.

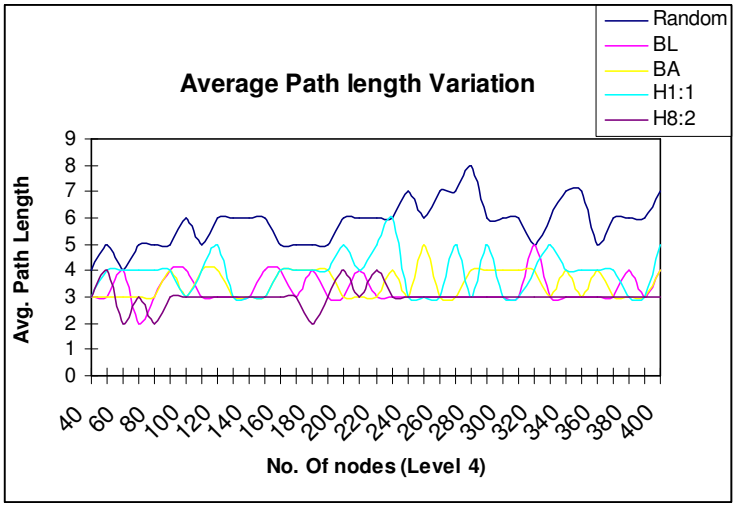

Figure 4: The variation of average path length followed by a query when the number of nodes in layer 4 varies from 40 to 400, number of nodes in layers 1,2 and 3 are constant, being 1,4 and 40 respectively.

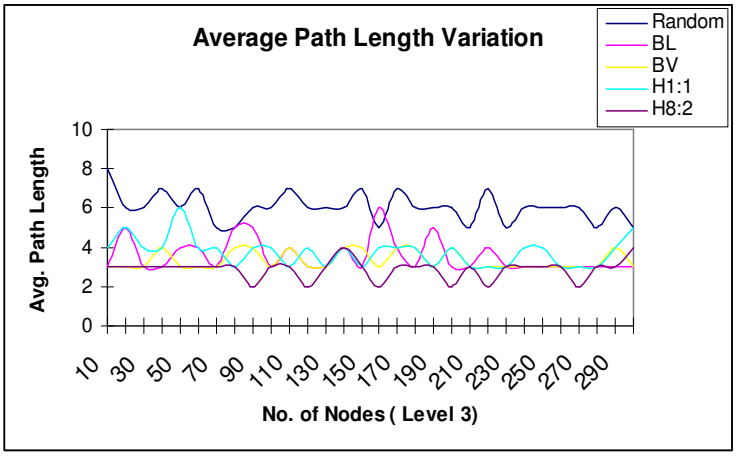

Figure 5: The variation of average path length followed by a query when the number of nodes in layer 3 varies from 10 to 300, number of nodes in layers 1,2 and 4 are constant, being 1,4 and 400 respectively.

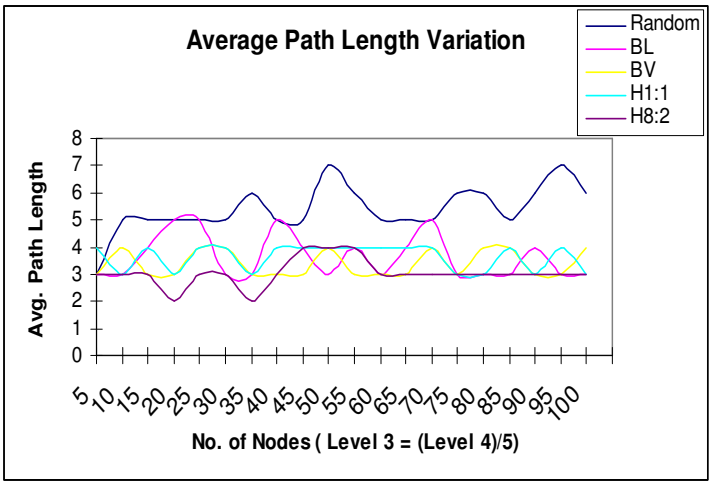

Figure 6: The variation of average path length followed by a query when the number of nodes in layer 3 and 4 varies from 5 to 100 and 25 to 500 respectively, number of nodes in layers 1 and 2 are constant, being 1 and 4 respectively.

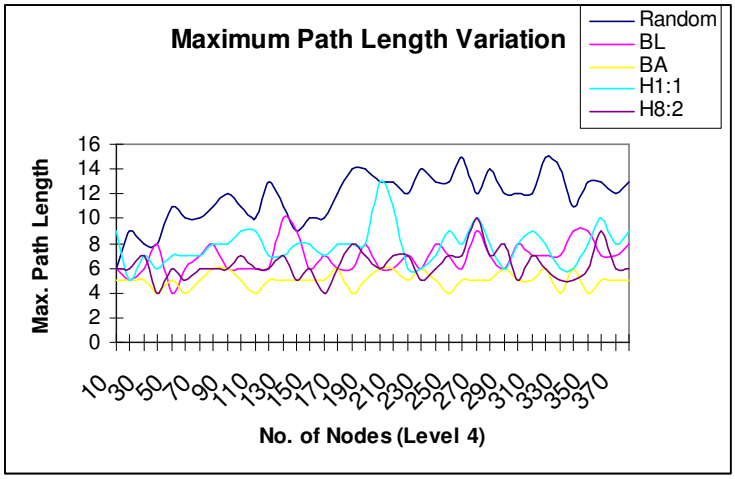

Figure 7: The variation of maximum path length followed by a query when the number of nodes in layer 4 varies from 40 to 400, number of nodes in layers 1,2 and 3 are constant, being 1,4 and 40 respectively. This gives an approximation of the worst case performance in terms of delay in the network.

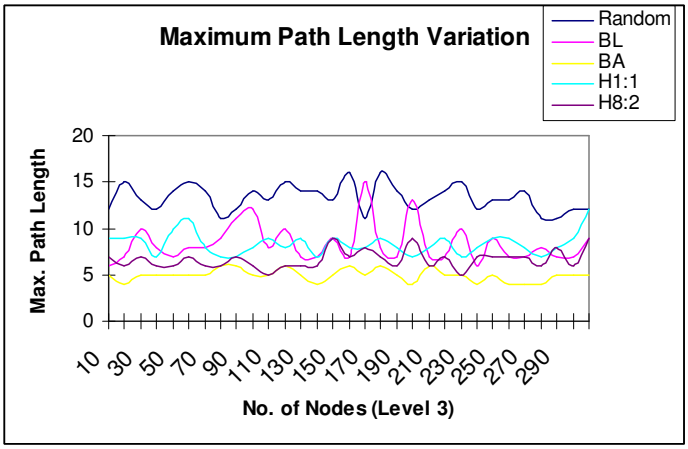

Figure 8: The variation of maximum path length followed by a query when the number of nodes in layer 3 varies from 10 to 300, number of nodes in layers 1,2 and 4 are constant, being 1,4 and 400 respectively.

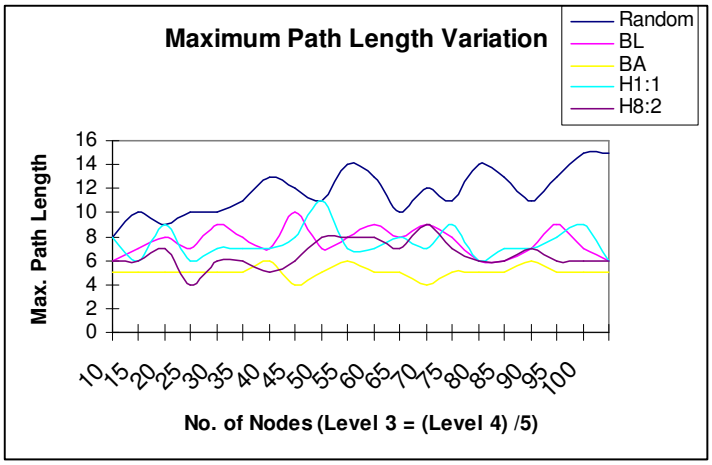

Figure 9: The variation of maximum path length followed by a query when the number of nodes in layers 3 and 4 vary from 5 to 100 and from 25 to 500 respectively, number of nodes in layers 1 and 2 are constant, being 1 and 4 respectively.

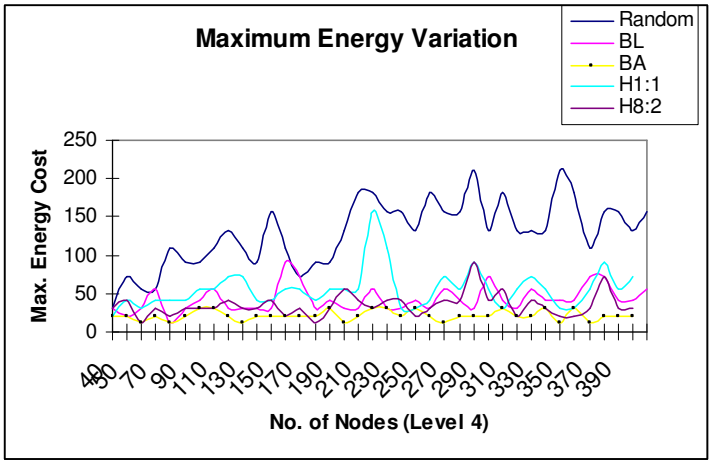

Figure 10: The variation of maximum energy expended by a query when the number of nodes in layer 4 vary from 40 to 400, number of nodes in layers 1,2 and 3 are constant, being 1,4 and 40 respectively.

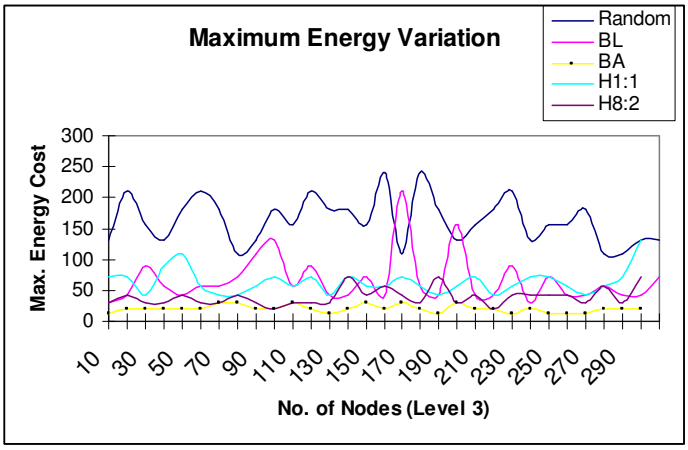

Figure 11: The variation of maximum energy expended by a query when the number of nodes in layer 3 vary from 10 to 300, number of nodes in layers 1,2 and 4 are constant, being 1,4 and 400 respectively.

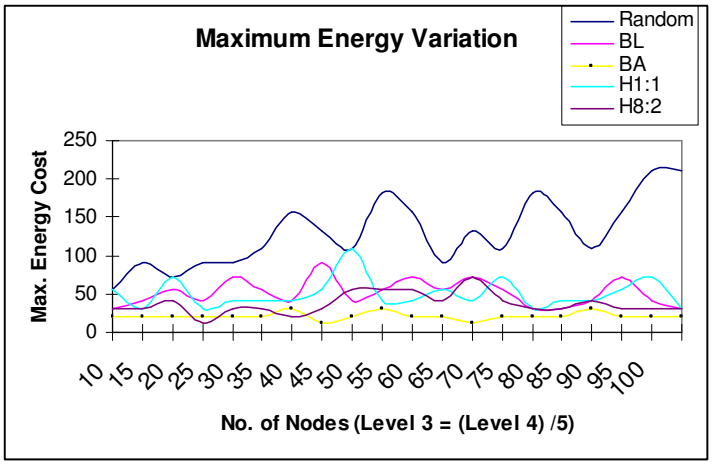

Figure 12: The variation of maximum energy expended by a query when the number of nodes in layers 3 and 4 vary from 10 to 100, and 50 to 500 respectively, number of nodes in layers 1 and 2 are constant, being 1 and 4 respectively.

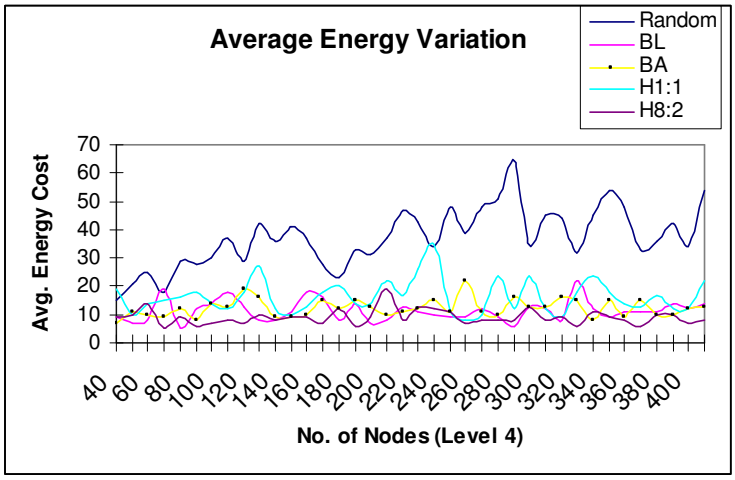

Figure 13: The variation of average energy expended by a query when the number of nodes in layer 4 vary from 40 to 400, number of nodes in layers 1,2 and 3 are constant, being 1,4 and 40 respectively.

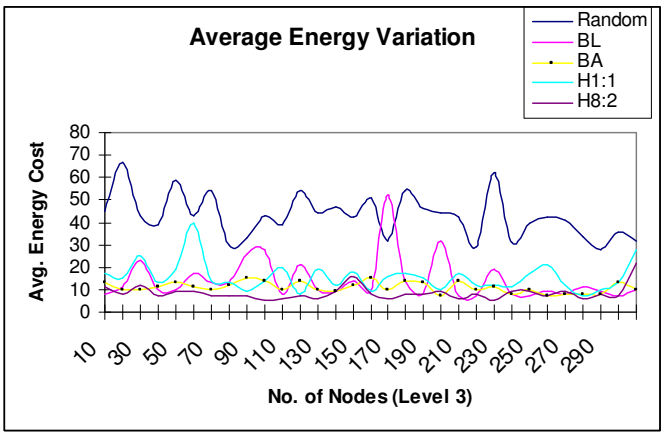

Figure 14: The variation of average energy expended by a query when the number of nodes in layer 3 vary from 10 to 300, number of nodes in layers 1,2 and 4 are constant, being 1,4 and 400 respectively.

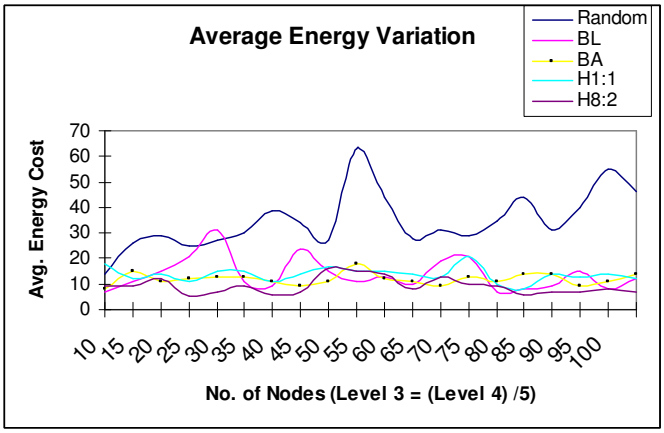

Figure 15: The variation of average energy expended by a query when the number of nodes in layers 3 and 4 vary from 10 to 100 and 50 to 500 respectively, number of nodes in layers 1 and 2 are constant, being 1 and 4 respectively.

The following table clearly indicated that the hybrid8:2 scheme is a good balance between the optimal attribute only based BA scheme and the link quality based BL scheme. The values in the table represent Standard deviations in the values measured by the simulations, The Hybrid8:2 scheme is the one that is closest to BA. In all the experiments conducted the BA scheme was definitely the optimal, which is quite natural since it bounds the subtrees which do not need to be probed for the query in question. However a significant observation is that a hybrid scheme which tries to construct the SRT based on equal weightage for both link metrics and attribute values is not very good in terms of both power depletion as well as path lengths that an actual query has to traverse. However we see that when the weights assigned to path lengths and attribute values are considerably skewed about 8:2 the hybrid scheme is much better than the Hybrid1:1, and not far away from the optimal BA scheme. The BL scheme at times seems better than the Hybrid 8:2 however on the whole, throughout the complete range of simulations where nodes have been varied from a 1:4:40:400 format in four levels to all sorts of combinations, attempting to change density of nodes drastically as well as gradually, we find that the Hybrid8:2 is closest to BV, Some of the closest values have been highlighted in the table below using pointers. From our readings we have almost 94% correlation between the MV and Hybrid8:2 scheme, the highest among all other schemes with the BV.

Standard Deviation	Random	BL	BA	Hybrid1:1	Hybrid8:2
Avg. Path Len. Level 4	0.91339	0.554306	0.556866	0.798291	0.402694
Avg. Path Len. Level 3	0.752058	0.850047	0.486373	0.716923	0.499462
Avg. Path Len. Level 3&4	0.88258	0.812728	0.502625	0.48936	0.510418
Max. Path Len. Level 4	2.038917	1.260235	0.655429	1.545194	1.206019
Max Path Len. Level 3	1.453953	2.078047	0.657463	1.176837	1.039024
Max. Path Len. Level 3&4	1.970172	1.187656	0.510418	1.277333	1.147079
Max. Energy Level 4	43.31827	16.93667	5.967432	25.2976	15.21931
Max. Energy Level 3	37.33717	39.61948	5.908987	20.12199	13.80229
Max. Energy Level 3&4	44.57991	17.252	4.691538	19.60961	14.16445
Avg. Energy Level 4	10.5822	3.765249	3.190156	5.85238	2.690799
Avg. Energy Level 3	9.837715	9.69913	2.359151	6.591784	3.324397
Avg. Energy Level 3&4	11.32429	6.417698	2.394621	2.876401	3.172248

6. References

[1] G. Pottie and W. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51 – 58, May 2000.

[2] J. Hill, R. Szewczyk, A.Woo, S. Hollar, and D. C. K. Pister. System architecture directions for networked sensors. In *ASPLOS*, November 2000.

[3] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler. Wireless sensor networks for habitat monitoring. In *ACM Workshop on Sensor Networks and Applications*, 2002.

[4] A. Cerpa, J. Elson, D.Estrin, L. Girod, M. Hamilton, , and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, 2001.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *MobiCOM*, Boston, MA, August 2000.

[6] S. Madden and M. J. Franklin. Fjording the stream: An architechture for queries over streaming sensor data. In *ICDE*, 2002.

[7] P.Bonnet, J.Gehrke, and P.Seshadri. Towards sensor database systems. In *Conference on Mobile Data Management*, January 2001.

[8] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. In *SIGMOD Record*, September 2002.

[9] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *OSDI* 2002.

[10] A. Woo. And D. Culler,"A transmission control scheme for media access in sensor networks." ACM MOBICOM, July 2001.

[11] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. *The design of an acquisitional query processor for sensor networks*. In Proc. of ACM SIGMOD, 2003.

[12] Y. Yao & J. Gehrke. Query Processing for Sensor Networks. In CIDR 2003.

[13] Niki Trigoni, Yong Yao, Alan Demers, Johannes Gehrke, and Rajmohan Rajaraman. WaveScheduling: Energy-Efficient Data Dissemination for Sensor Networks. In Proceedings of the International Workshop on Data Management for Sensor Networks (DMSN 2004) held in conjunction with VLDB 2004.
[14] Sumi Helal, Nitin Desai, Varum Verma and Choonhwa Lee, "Konark – A Service Discovery and Delivery Protocol for Ad-hoc Networks," Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, March 2003.
[15] Christian Bettstetter and Christoph Renner. A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol. In *Proc. EUNICE Open European Summer School (EUNICE)*, Twente, Netherlands, Sept 13-15, 2000.
[16] Xiaoyan Hong and Mario Gerla , Dynamic Group Discovery and Routing in Ad Hoc Networks **-** - *Proceedings of the First Annual Mediterranean Ad Hoc Networking Workshop (Med-hoc-Net 2002), Sardegna, Italy, Sept. 2002.*
[17] http://www.unet.univie.ac.at/aix/aixprggd/progcomc/xns_seqprot.htm
[18] http://www.chipcon.com/index.cfm?kat_id=3&action=faq&type=show_all
[19] http://www.media.mit.edu/resenv/pubs/papers/2003-12-circuit_cellar5.1.pdf