

Response Based Classification of Sensor Query Networks

Ömer Sinan Kaya

Sebnem Baydere

Department of Computer Engineering Yeditepe University,

Kayisdagi, Istanbul, Turkey

oskaya@softhome.net

sbaydere@cse.yeditepe.edu.tr

ABSTRACT

Sensor networks use densely distributed micro-sensors with low power communication capabilities to collect data from the area of interest. Queries are disseminated and data is collected at aggregation nodes for processing. In this paper, sensor networks are classified according to the response characteristics of the query they are formed to process. A novel multi-tier architecture, comprised of mobile aggregation nodes and static sensors, is used as the basis of analysis. A reverse multicast tree network is dynamically formed among an aggregation node and its associated sensors for response delivery. Traffic loads of query types are analyzed and some optimization methods are presented to reduce the message delivery rate of some types. Experimental results of implementation for TinyOS running mica sensor hardware are also given.

Keywords: Sensor network, query processing, TinyOS

1. Introduction

Wireless sensor networks are environmental monitoring systems that are composed of densely deployed micro sensor nodes operating in an untethered and unattended mode. The main goal to construct a sensor network is to detect events occurring in the area of interest. Each node is designed to monitor the environment for events specified in the queries driven by the applications or the events specified by the deployer. Collaboration among hundreds to several thousands of limited computation, communication and sensing capable sensors is needed for information retrieval in the area of interest. Several design challenges exist for intelligent query processing systems as well as networking of such devices. Most of the designs incorporate Mobile Ad Hoc Networks as in [7] and [1] to allow easy access of

disseminated data as in Figure 1. Our experiences in the design of an architectural model for environmental monitoring proposed in [1] have shown that general classifications of such networks will be useful in attacking these problems. In the multi-tier architecture, a fully blown query processing system was designed to map user defined high level queries to simple sensor queriers. Mechanisms were defined for query resolution, dissemination and data collection at the intermediate components. Thus, in this paper, we present a generalized query classification approach that can be used to analyze the sensor network performance in terms of traffic load.

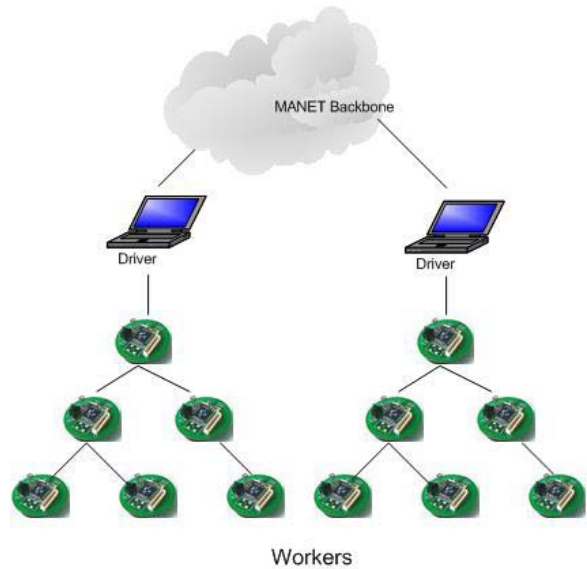


Figure 1: A Typical Query Processing Architecture

The rest of the paper is organized as follows. Section 2 explains the querying network model in general. Section 3 discusses response based query types. Section 4 explains response generation characteristics and an optimization method. Section 5 gives implementation experiments and

Section 6 contains concluding remarks.

2. The Querying Network Model

The novel sensor network model is comprised of a collection of n -hop trees tied together by an ad hoc mobile backbone of root nodes acting as intermediaries. We call root nodes as sensor *drivers* and their associated sensors as *workers*. Each sensor network can be perceived as a reverse multicast tree formed to process one query at a time. Query time period (*QTP*) is limited and workers may fail between queries with independent probabilities. Workers get a query and a set of attributes to run the query. In our model processing method is determined by the query type. Query type is used to instruct the sensor query processor. It is assumed that sensors are being scattered around the terrain and operate low power radio for transmission to get and respond queries. The monitoring application may have two main objectives for querying; obtain readings of the sensing units in the area of coverage or get/set constant values found on the nodes such as battery level, node id etc. First classification is whether the query is *sensing related* or *node related*. Considering that, for sensing related queries single raw data readings may not give enough idea about the environment, to compile data a number of raw samples may be needed. Therefore the querying model encodes how many samples (*SC*) must be taken and how frequent this readings has to be obtained from the sensing interface.

2. Response Based Query Types

General purpose query processing systems may handle a number of different query types. Query type specifies “*how data should be retrieved from a node*” and “*when a response packet should be generated*”. Three attribute classes are defined:

- **Response attempt** is either continuous or one shot
- **Response generation method** is either simple or complex
- **Response transmission** is either concatenated or immediate

Descriptions of query attribute types are given below:

1. **Continuous Queries** last for query timeout period and may result in multiple response packets from sensor nodes. These queries are used for periodic data collection. Example: *Report daily temperature for a week at most 3 hops away.*
2. **One shot Queries** result in a single response packet from every sensor node. These queries are used to receive state information one compiled result from

sensors. Example: *Report temperature readings from all nodes.*

3. **Simple Queries** return raw data readings from sensors without applying any compilation function. Example: *What is the battery level? Or is sensing interface on?*
4. **Complex Queries** contain a compilation function such as sum, total, min, max of samples, to be applied to a number of raw sample readings Example: *Return temperature reading if after taking 3 samples at 3 min frequency any sample value is greater than 100 degrees.*
5. **Aggregate (concatenated) Queries** delay packet transmission at intermediate nodes during response delivery. Nodes keep responses from their children until their send buffer is full or timer expires. One network packet containing all responses in the buffer is transmitted. A typical query example which may benefit from this type of aggregation: *What is the number of nodes in the area?*
6. **Non-Aggregate (immediate) Queries** return each response separately. As soon as a packet is generated or received by the node, it is relayed towards the root. Example: *Return immediate response if at least one reading of 5 samples taken at 1 second frequency is over 70 degrees.*

Sensor networks are classified with three binary digits encoding their query attributes. For example: **010** type network processes a *one shot, complex* and *immediate* query. This network generates zero or one response from a simple reading on each connected sensor, resulting at most $N-1$ network packet to be forwarded towards the root. Additionally, each data packet is relayed immediately via intermediate nodes. Figure 2 illustrates query driven network types’ classification. As a matter of fact, a sensor network is formed for a limited period of time to process one *network query*. Each sensor in the network is instructed to compile query readings at given frequencies. *Response generation frequency* is defined as a function of *sample count* (*SC*) and *query frequency* (*QF*) values received during setup in query parameters. A network query can be decomposed into a number of sub-queries. Compiled result of each sub query is locally buffered and one network packet is generated at *RGF* intervals for all sub query responses.

$$RGF = Max(SC).QF \quad (1)$$

As the *response generation method* (*RGM*) may vary for each sub query $S(i)$, traffic intensity of the network as a function of network query type can only be analyzed for the best and worst case sub query type combinations. For example: simple queries generate one response at each node

whereas complex queries may result in zero or one response. If set of generation methods is defined as

$$RGM-SET = \{Simple, Complex\} \quad (2)$$

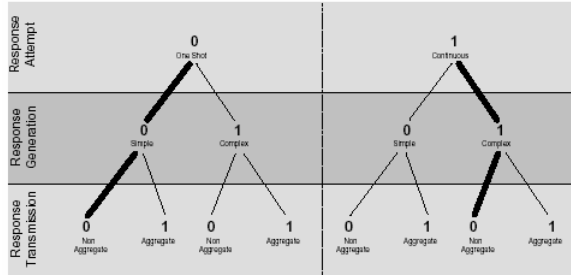


Figure 2 : Query Driven Network Types

Then the response generation method for the network query can be given as follows:

$$NetQuery_{RGM} = \bigcup_{i=1}^4 SubQuery(i)_{RGM} - \{Complex\} \quad (3)$$

This predicate statement indicates that if at least one sub query is of *Simple* type then network query RGM is also *Simple* which generates one response per node. If all sub queries are *Complex* then the network query type is also *Complex* generating zero or one response per node.

Continuous queries cause periodic processing on the nodes. The number of periods (QCycle) is found by dividing query time period (QTP) by response frequency (RGF) from eq.(1).

$$QCycle = QTP / RGF \quad (4)$$

Query Type	Response Range
00(0,1)	[N, N]
01(0,1)	[0, N]
10(0,1)	[(N*QCycle), (N*QCycle)]
11(0,1)	[0, (N*QCycle)]

Table 1 : Traffic Intensity in N-hop sensor network

From eq.(4) and network query type definition, minimum and maximum number of packets injected into the network can be estimated for network query types. Table 1 illustrates the range of number of responses generated in an n-node query network.

3. Response Generation

When a sensor node receives query parameters during setup it sets up a one shot query timeout timer to be fired at *QTP* and a continuous query frequency timer to be fired at each *QF* time. With each frequency timer tick, if the sub-query is sensing related relevant sensing unit is switched on, the result

is gathered and *SC* of the sub-query is decremented by one. This process continues until *QTP* is reached or *SC* of all sub queries reach zero which indicates that a response packet should be generated. If *Continuous flag* is not set, frequency timer is stopped after packet transmission in order to save energy by avoiding unused timer interrupts. In active state, query nodes either generate query data or relay others data. In either case, nodes propagate messages towards upper levels of the tree by transmitting packets to their parents. They learn parent-id and region-id during setup packet processing. Nodes process messages only if their local region-id and node-id is equal to the region-id and message destination-id in the received packet. This allows simultaneous queries to be run in the same physical region without query networks interfering each other.

Since we assume simultaneous sub queries in one network query, sub query responses have to be synchronized with each other on the sensor node. Assuming different sample counts to be set up for different sub queries, nodes wait for all sub queries to be resolved in order to generate a response message. A single RGF is set for the whole network. With every frequency timer tick, it is checked if all sub query sample counts have reached zero or not. Completed sub queries switch of their sensing interface and goes to sleep. When all sub queries are completed a response message is generated. RGF periods are given in Figure 3. The Figure illustrates a continuous network query which has four sub queries with different sample counts. Query instances are scheduled at *t4*, *t8* etc. In order to schedule next instance all sub query samples for the previous instance have to be collected. At this point a response message is posted and all sub queries are rescheduled by initializing their number of sample counts to their initial values.

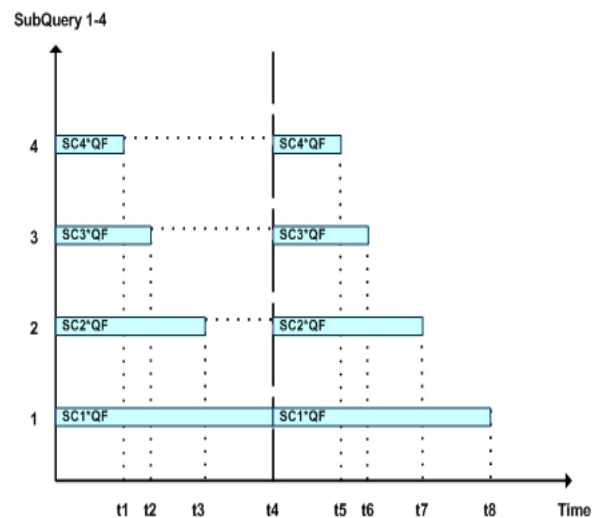


Figure 3 : RGF Periods Time Gantt chart

3.1 Message Time Synchronization

A response message might carry a single node response or responses from multiple nodes. Thus, the response message is a variable length message. Each response data carries its originator's hop count to specify how many hops away this event has occurred. This way, driver node can have a rough idea about the location of the event in terms of transmission range and hop count. Since not all of the nodes participating in query are required to match all sub queries, some nodes joining the query can be simple message forwarder. So, how many of the requested sub queries can a node answers to has to be specified. Sensor networks are mostly known as unreliable and there is no guarantee about the order of arrival of the events to the driver node. Therefore, response messages about the same event initiated by neighboring nodes might arrive at different times to the driver. While some packets might be lost on the way, others might arrive out of order or some nodes might lose contact with their local parents and change return path dynamically. As a result, the driver node is expected to discriminate these messages from each other. In the case of a target tracking application, messages are required to be sorted by time to provide target path to the monitoring application. In order to distinguish events from each other a timing mechanism is required. Recent research groups [8], [9] have used messaging based synchronization primitives but message transmission is the most costly operation on sensor nodes and it's not desirable for each node to transmit message to each other to keep synchronized. Proposed query processing algorithm provides its users with relative message delivery time so that client nodes can have a rough idea about the occurrence of event. Each response message has a response time field which is used for time stamping. Each time a response message is processed on each node at the forwarding path its time count is incremented by the amount of time difference between message reception and message transmission and by a theoretical propagation delay calculated by using 12kbps transmission throughput and packet length. Therefore, messages for the same event arriving via a longer path will have relatively greater time. Driver node can filter the events that have already been delivered to client node so that delayed packets are also taken care of.

3.2 Immediate vs. Aggregated Delivery

Two different response generation methods can be used. Concatenation flag in the setup is used to switch between modes.

Immediate Delivery can be used for critical applications like a fire in the forest or a chemical attack in the war, events

should be delivered to driver nodes immediately. With concatenation flag unset, sensor nodes forward their messages to their parent nodes without delaying.

Aggregated Delivery can be used in order to save energy. Messages that are carried over the broadcast network are concatenated at upper level nodes so that the number of transmitted packets is reduced. Since each packet has an identifier of the responding node, concatenation is more efficient in terms of number of bytes transmitted.

For example, in our prototype model each node which responds to a query message with one sub query will send a response message with 37 bits of message header for identifying message type, destination, region id and response count and 41 bits for response data, resulting in 78 bits.

```
ProcessResponseMessage()
{
    TOS_Msg msg;
    if (ReceiveMsg(&msg)) begin
        if (EnoughPlaceExistsInLocalResponseMessage())
            CopyMessageToLocalResponse(msg)
        else begin
            CopyPartOfMessageUntilPacketFull(msg, local_response_message)
            TransmitMessage(local_response_message)
            AllocateBuffer(&local_response_message)
            CopyRemainingsOfMessage(msg, local_response_message)
        end
        if (PacketFull(local_response_message))
            TransmitMessage(local_response_message)
    end
    if (local_response_message_ready) begin
        if (EnoughPlaceExistsInLocalResponseMessage())
            AppendLocalResponseMessageToLocalResponseMessage()
        else begin
            TransmitMessage(local_response_message)
            AllocateBuffer(&local_response_message)
            RetryLocatingLocalResponse()
        end
        if (PacketFull(local_response_message))
            TransmitMessage(local_response_message)
    end
    if (Expired(Concatenation_timer)) begin
        if (ThereIsResponseMessageNotSent())
            TransmitMessage(local_response_message())
    end
end
}
```

Figure 4 : Response Aggregation Algorithm

When the node is configured with the concatenation mode, it will put 4 messages with 41 bits of response data and a header to the packet instead of transmitting 78 bit response data four times resulting in 55% gain. Response Aggregation algorithm is given in Figure 4. One drawback of concatenation is the latency for response messages. With concatenate flag set, sensors wait for a *concat timeout* interval to fill the packet with the responses from other nodes.

Eq. (5) gives timeout for a node to relay its message. It is proportional to the level of the node in the tree ($fd-hc$) and node's neighbor count (nc).

$$T_{concat} = RGF.(fd - hc).nc \quad (5)$$

4 Experiments



Figure 5: Mica Hardware

Sensor node software is developed using TinyOS operating system [2] and NesC [3] programming language on MICA2Dot hardware as shown in Figure 5. TinyOS simulator, TOSSIM [4] has been used to test algorithms for scalability. TOSSIM is a discrete event simulator that allows code written for TinyOS to be used for simulation directly. Primary goal of TOSSIM simulator is to emulate how TinyOS works on motes by using the same components of TinyOS and changing behavior of system dependent code. For example TOSSIM interrupts do not preempt each other and tasks are not interrupted by TOSSIM interrupts either. TOSSIM has been used extensively to test the algorithms against certain inputs. Debug messages have been used to trace how applications perform. Some applications have been developed that use TOSSIM sockets to inject setup messages and collect response messages have also been developed.

TOSSIM uses ADC and RFM models to simulate different communication scenarios and different ADC readings that can be achieved. Lossy model is used to model node connectivity with graph edges and loss rates for that link. We have used a customized version of lossy model for our tests. According to [4] TOSSIM uses mica communication stack at bit level simulation so that MAC protocols also effect how messages are transmitted. However, our tests proved that simulation of radio communication is unstable and results in higher rates of messages getting lost as well as some messages not being delivered because of inaccurately designed TinyOS Active Message Acknowledgement methodology [4]. Additionally, Mica2 stack and other MAC protocols such as SMAC [5] are proved to be much stable

than MICA stack and we didn't want to base our responses to a specific kind of MAC protocol since our model is independent of the underlying MAC Protocol. As a result, we have replaced simulator's bit level communication stack with packet level communication and then utilized packet delivery with bit error rates from lossy model following equations give the formulas we used to compute packet error rates resulting from medium. Since equation contains packet length, packet loss probability is different for various sizes of packets.

$$P(\text{NoBER}) = 1 - P(\text{BER}) \quad (6)$$

$$P(\text{NoPER}) = P(\text{NoBER})^{\text{packet length}} \quad (7)$$

We have also modeled losses resulting from MAC layer collisions. In the equation, N is the number of neighboring nodes in the same transmission range with transmitting node and α is the probability that at least two nodes will transmit a packet at the same time after detecting the channel as clear. We have used α as 0.05 in our experiments.

$$P(\text{Col}) = \sum_{i=2}^n C_n^i \cdot \alpha^i \cdot (1 - \alpha)^{(n-i)} \quad (8)$$

The probability that a transmission succeeds is the probability that there is no bit error rate and no collision in the channel.

$$P(\text{TX success}) = P(\text{NoPER}) \cdot P(\text{NoCOL}) \quad (9)$$

Simulation Parameters	Values
Terrain Size	200m x 200m
Transmission Range	30m
Sensing Range	30m
Simulation Time	300 simulation seconds
Alpha	0.05
BER	0.00001
# of Generated Events	80
# of Nodes Used	25,50,100
# of Generated Topologies per node number	10
Number Of Tests Repeated for each topology	30
Event Duration	10-15 seconds

Table 2: Simulation Parameters

From here, packet loss probability (PLP) for the transmission of a packet is calculated as the 1's complement of successful packet transmission probability.

Prior to simulation with TOSSIM, we have created simulation parameters offline and inspected the results from the TOSSIM simulation. A trace generator tool has been used to create events and position nodes in the terrain.

Constants used in the simulation can be seen on Table 2.

We used a terrain size of 200 meters x 200 meters and a transmission range and a sensing range of 30meters. In order to use results from these output files, we have written a new ADC model. This ADC model reads the output files and fills the necessary data structures in the simulator for node positions and lossy model parameters.

Event generator tool creates events in the terrain to last for some duration at random positions and startup times. When the simulation is started with the our ADC model, simulator checks for each node if the event is in the sensing range of the node and creates a simulation event to take place at event start time away from setup message reception time and removes the event after event duration time has passed. During this time, if a node tries to gather data from ADC it detects the event.

Query parameters used for simulation can be seen on Table 3. We have used a continuous query message with 60 minutes QTP, 5 seconds QF. Each node tries to read a value lower than 65535 every 5 seconds and when new ADC model is used, it returns 1 if it detects an event registered for the node; otherwise it returns 65535. So a message will be transmitted by the node if it detects an event. We have made tests for 25, 50 and 100 nodes with injecting the same query for all tests to see how well the model performs. Additionally, we have generated 10 different topologies for each node number. We have made 30 tests for each topology and calculated the average return values for these tests. We have also created a simple battery simulation environment for the simulator. After the simulation is started, each node is assigned a battery level parameter as 3000mAh which is taken from a standard battery. At each packet transmission and reception, this value is decremented by transmission and reception powers taken from [6] respectively.

Query Parameter	Value
Flooding Degree	20
Region ID	1
Continuous Flag	True
Query Timeout	60 Minutes
Query Frequency	5 Seconds
Sub query Count	1
Compilation Function	LOWER
Compilation Data	65535

Table 3: Query Parameters

Sleep energy of 30uA taken from [7] is decremented between each successive simulation events. During packet transmission, the battery level of the node that data is to be transmitted to is checked and if it is below 0, no packet is transmitted to that node which means that node has run out of power.

4.1 Immediate Response Tests

With this experiment setup, we have measured connectivity rate, event delivery rate and message delivery rate for different number of nodes. Test results can be seen in Figure 6, Figure 7 and Figure 8 respectively. Connectivity is the ratio of the number of nodes joining query processing and total number of nodes in the terrain. When Figure 6 is examined, it can be seen that higher number of nodes result in higher rates of connectivity since node density in each region becomes higher as node count increases. As a result, nodes receiving setup message increase as well.

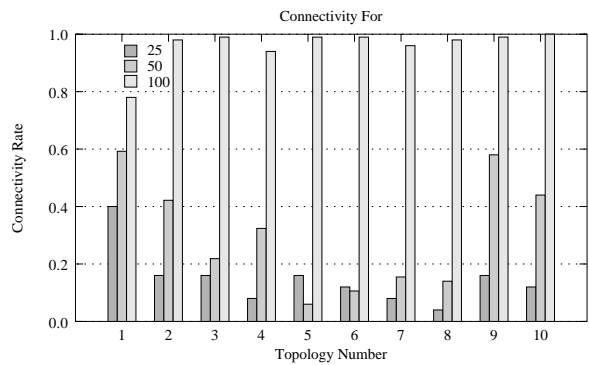


Figure 6 : Connectivity Ratios for Immediate Response Delivery

Event delivery ratio is the rate of number of events delivered to monitoring application to total number of events registered for all nodes in the terrain to be detected. It can be seen from Figure 7 that as the number of nodes increase, the number of events delivered to root node increases as well. This is because in the case of lower node density, events will be registered for node to be detected but as some nodes will not receive setup message event will be missed.

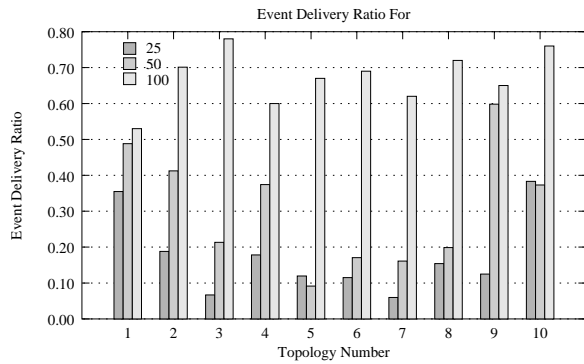


Figure 7 : Event Delivery Ratios for Immediate Response Delivery

Message delivery ratio is the rate of messages delivered to the root to the total number of messages processed in the network. It can be inferred from Figure 8 that as the number of nodes increase, the number of messages successfully delivered decrease. Since node density increase with increasing node number, probability of message not being delivered will increase as well. PLP of 0.10 has been seen with 100 nodes during simulation.

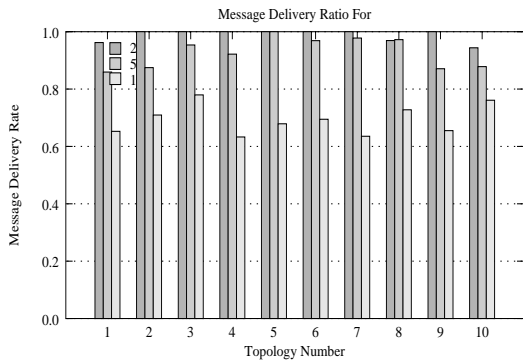


Figure 8 : Message Delivery Ratios for Immediate Response Delivery

4.2 Concatenation Tests

We have repeated same tests for concatenated mode of delivery. The simulation parameters at Table 2 remained same except concatenate flag. Similar results have been achieved in connectivity tests applied for concatenated response delivery in Figure 9. Higher number of nodes result in better connectivity since the number of nodes receiving setup packet will hear will increase as node density is increased in the area.

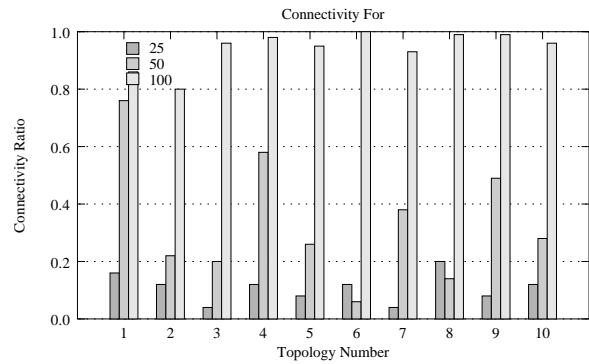


Figure 9 Connectivity Ratios for Concatenated Response Delivery

As with the case of connectivity, similar results have been achieved with event delivery as well. According to Figure 10 tests with higher number of nodes resulted in better event delivery rates because events are randomly generated in the terrain. As node density increases, probability of having a node close enough to detect events increases and since node number is high, most of the nodes will be connected and detected events will be delivered to root.

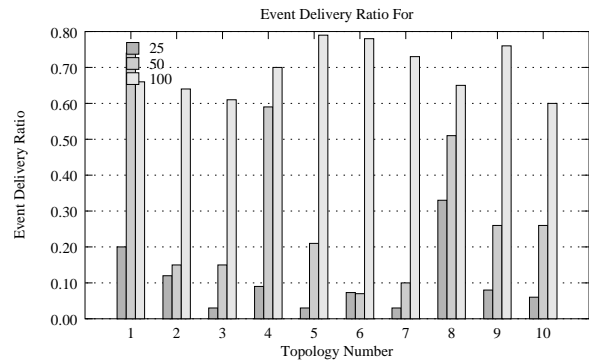


Figure 10 : Event Delivery Ratios for Concatenated Response Delivery

When Figure 11 is inspected, message delivery rate seems to be quite low. However, lower message delivery rate means that lower number of messages is transmitted in the terrain to detect events. Since response messages are concatenated, one message carries at a maximum of four response messages. Therefore, event delivery rates of 80% has been achieved with 100 nodes by only transmitting 0.2 of total messages while similar results with immediate delivery can only be achieved with transmitting 0.78 of all messages. Another advantage of concatenation is that as node number increases number of messages that must be delivered decreases while achieving higher levels of event delivery. This indicates that concatenation is mostly suitable for dense networks.

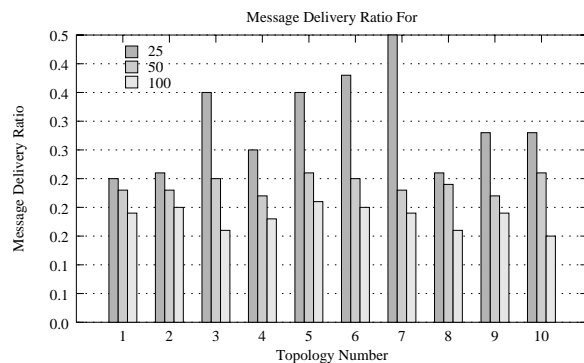


Figure 11 : Message Delivery Ratios for Concatenated Response Delivery

5. Conclusion

Trying to network a large number of low power sensor nodes is a challenging problem that has recently been the focus of a growing research effort [2, 10, and 11]. Particularly data routing, node addressing and support for different classes of services are the primary issues tackled at the network layer. Traditional ad hoc networking protocols are generally not resource-aware or resource-adaptive. On the other hand, conventional data dissemination approaches like flooding and gossiping waste valuable communication and energy resources by sending redundant information throughout the network. Therefore, these schemes have been difficult to adapt, and as a result many new algorithms have been developed.

Generalized query models may affect the efficiency of underlying power aware communication protocols. We have discussed a model for query classification in sensor networks based on response generation. A simple response traffic analysis is given and experiments with the implementation of a query processor on TinyOS running mica2 and micadot hardware are discussed.

6. Acknowledgment

The authors gratefully acknowledge the group members Mesut Ali Ergin, Özlem Durmaz, Sinan Buyruk, Metin Koç and Onur Ergin for several useful discussions on the design of the overall architecture. This work is partially supported by The Scientific and Technical Research Council of Turkey under grant 103E001/EEEAG-AY-44

References

[1] S. Baydere and M.A. Ergin, "An architecture for service access in mobile ad hoc networks", Proc. IASTED WOC (Banff Canada, July 2002) 392-397.

[2] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors", In Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (Cambridge, MA, USA, November 2000) 93-104.

[3] D.Gay, P.Levis, R.von Behren, M.Welsh, E.Brewer and D.Culler, The Nesc Language: "A Holistic Approach to Networked Embedded Systems"

[4] P.Levis, N.Lee, M.Welsh and D.Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Application", Sensys'03

[5] W. Ye, J.Heidermann and D-Estrin, "Medium Access Control with Coordinated, Adaptive Sleeping For Wireless Sensor Networks" USC/ISI Technical Report ISI-TR-567(January 2003)

[6] "MPR- Mote Processor Radio Board MIB- Mote Interface / Programming Board User's Manual" Crossbow Technology, Inc. San Jose, CA (October 2003)

[7] A. Mainwaring, et al, "Wireless Sensor Networks For Habitat Monitoring" WSNA'02 Atlanta, Georgia, USA, September 28 2002)

[8] Richard Karp, Jeremy Elson, Deborah Estrin, and Scott Shenker, "Optimal and Global Time Synchronization in Sensornets", CENS Technical Report 0012, April 10, 2003.

[9] J. Elson and K. Rømer. Wireless Sensor Networks: "A New Regime for Time Synchronization." ACM SIGCOMM Computer Communication Review (CCR), 33(1):149-154, January 2003.

[10] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM '00), August 2000.

[11] D.Ganesan, R.Govindan, S.Shenker, D.Estrin: "Highly Resilient, Energy Efficient Multipath Routing in Wireless Sensor Networks", ACM Mobile Computing and Communications Review, Vol.5, No 4 (October 2001)