International Journal of Computer Science and Network (IJCSN)
Volume 1, Issue 3, June 2012 www.ijcsn.org ISSN 2277-5420

RELIABILITY ORIENTED TRANSPORT PROTOCOL IN WSN

[1]Bejoy.B.J, [2]Dr.B.Paramasivan

[1]Asst.Professor, Department of CSE, P.S.R Engineering College, Sivakasi,
Virudhunagar dist, Tamilnadu, India

[2]Prof&Head, Department of CSE, National Engineering College, Kovilpatti,
Thoothukudi dist, Tamilnadu, India

Abstract

Wireless sensor network is a special form of wireless networks dedicated to surveillance and monitoring applications Reliability in wireless sensor network is application specific. The specific form of reliability might change from application to application. Our idea is to generate reliability based transport protocol that is customizable to meet the needs of emerging reliable data applications in sensor networks and is also adaptive when the nodes are mobile. In our approach, clusters are formed for minimizing energy dissipation. The nodes maintain a neighbor list to forward data and any changes in the local topology can trigger updates to a node's neighbor list. If a node notices that its neighbor list has changed, it can spontaneously re-advertise all of its data thus providing reliable transport in mobility conditions also. Our approach has five phases-setup, relaying, relay initiated error recovery, selective status reporting and node supervising. Our simulation results prove that the proposed approach can outperform existing related techniques and is highly responsive to the various error and mobility conditions experienced in sensor networks.

Keywords: *Mobility, Re-tasking, Reliability, Transport protocol*

1. Introduction

Wireless Sensor Network consists of tens or thousands of sensor nodes scattered in a physical space and one or more Base stations or Sinks. Even thought developed for military applications now they find a wide variety of civilian applications also. Some of the applications are Target tracking, Animal monitoring, Vehicle monitoring. The need (or lack thereof) for reliability in a sensor network is firmly dependent upon the specific application the sensor network is used for. Some applications like re-tasking or reprogramming sensor nodes [upgrading software or algorithms, adding codes, scripts etc] over -the-air requires assured delivery of high-priority events to sinks. We believe that as the number of sensor network applications grows, there will be a need to build more powerful general-purpose hardware and software environments capable of reprogramming or retasking sensors to. These general-purpose sensors would be capable of servicing new and evolving classes of applications. Unlike traditional networks [e.g., Internet protocol (IP) networks], reliable data delivery is still an open research question in the context of wireless sensor networks.

The vast majority of sensor network applications do not require reliable data delivery. For example, in applications such as temperature monitoring or animal location tracking, the occasional loss of sensor readings is tolerable and, therefore, the complex protocol machinery that would ensure the reliable delivery of data is not needed. Directed diffusion [1] provides robust dissemination through the use of multipath data forwarding, but the correct reception of all data messages is not assured. Data that flows from sources to sinks is generally tolerable of loss. On the other hand, data that flows from sinks to sources for the purpose of control or management (e.g., retasking sensors, actuation) is sensitive to message loss. Due to the application-specific nature of sensor networks; it is hard to generalize a specific scheme that can be optimized for every application. Rather, the focus of this paper is the design and evaluation of a new transport system that is simple, robust, scalable, and customizable to different applications' needs and can work well in mobile environment like that of wireless sensor network. Ours is a simple approach with minimum requirements on the routing, minimum signaling, thereby reducing the communication cost for data reliability, and finally, responsive to high error rates allowing successful operation even under highly error-prone conditions. Due to the application-specific nature of sensor networks, it is hard to generalize a specific scheme that can be optimized for every application.

2. Methodology

The proposed methodology will satisfy scalability and robustness. It's scalable because it supports minimum

signaling thereby reducing communication cost for data reliability. It is robust because it is responsive to a wide range of operational error conditions. It can be used in conditions where there is mobility of node, which is a main concern in PSFQ [2]. We have taken a different approach in comparison to traditional end-to-end error recovery mechanismsin which only the final destination node is responsible for detecting loss and requesting retransmission. The biggest problem with end-to-end recovery has to do with the physical characteristic of the transport medium. Sensor networks usually operate in harsh radio environments, and rely on multihop forwarding techniques to exchange messages. Error accumulates exponentially over multihops, therefore, packet loss and reordering is more likely.

Our approach uses hop-by-hop error recovery in which intermediate nodes also take responsibility for loss detection and recovery so reliable data exchange is done on a hop-by-hop basis rather than end-to-end. The key idea that underpins the design of this protocol is to distribute data from a source node by pacing data at a relatively slow speed, but allowing nodes that experience data loss to fetch any missing segments from immediate neighbors very aggressively.

Messages that are lost are detected when a higher sequence number than expected is received at a node triggering the relay-initiated error recovery operation, i.e., an energy-efficient negative acknowledgment system that our protocol is based on. The motivation behind our simple model is to know what the immediate neighbors are and to pass the data to them even in mobile conditions.

3. Protocol Description

In Wireless Sensor networks the main constraint is availability of energy. So we are using clustering concept to enhance scalability and efficient communication. It's also an efficient method to lower energy consumption as higher energy nodes can be used to process and sent the information while low energy nodes can be used to perform the sensing in proximity of the target. Then we are setting up a neighbor list in each node so that it can know its immediate neighbors. This will help in broadcasting data to immediate neighbors and also help in sending NACK messages for error recovery.

First we are using the concept of clustering for energy efficiency and then using a neighbor list to broadcast data to immediate neighbors and also for error recovery during mobile conditions. In this proposed methodology, there are five phases-setup, relaying, relay initiated error recovery, selective status reporting and node supervising.

3.1 Setup phase

The first phase is the set up phase of LEACH algorithm [3]. In LEACH, the nodes organize themselves into local clusters, with one node acting as the cluster head. All non-cluster head nodes transmit their data to the cluster head, while the cluster head node receives data from all the cluster members, performs signal processing functions on the data (e.g., data aggregation), and transmits data to the remote BS. But being a cluster head node is much more energy intensive than being a noncluster head node. If the cluster heads were chosen and fixed throughout the system lifetime, these nodes would quickly use up their limited energy.

Once the cluster head runs out of energy, it is no longer operational, and all the nodes that belong to the cluster lose communication ability. Thus, LEACH incorporates randomized rotation of the high-energy cluster head position among the sensors to avoid draining the battery of any one sensor in the network. In this way, the energy load of being a cluster head is evenly distributed among the nodes. We are only incorporating the set up phase of LEACH i.e. to set up clusters and select appropriate cluster head. Each sensor elects itself to be a cluster head at the beginning of round $r+1$ (which starts at time t) with probability $P_i(t)$. $P_i(t)$ is chosen such that the expected number of cluster head nodes for this round is k . Thus, if there are N nodes in the network

$$E[\#CH] = \sum_{i=1}^{N} Pi(t) * 1 = k$$

The creation of cluster and assigning special tasks to cluster heads can greatly contribute to overall system scalability, lifetime and energy efficiency.

3.2 Message Relaying

Recall that our protocol is not a routing solution but a transport scheme. In a case where a specific node needs to be addressed directly, instead of a whole group of sensors, which is the norm, then our protocol can operate on top of existing routing schemes (e.g., DSDV [4]) to support reliable data transport. A user node can use time-to-live (TTL)-based, as well as group address filtering methods to control the scope of its retasking operation .Note however, that this method does not provide accurate scope control because in many cases the

intended receivers cannot be neatly defined by a limit of TTL. To enable local loss recovery and in-sequence data delivery, a data cache is created and maintained at intermediate nodes. The relaying operation is important in controlling the timely dissemination of code segments to all target nodes, and providing basic flow control so that the retasking operation does not overwhelm the regular operations of the sensor network. This requires proper scheduling for data forwarding.

3.2.1 Relay Timers*:*

A user node broadcasts a packet to its neighbors until all the data fragments have been sent out. Neighbors that receive this packet will check against their local data cache discarding any duplicates. If this is a new message our approach will buffer the packet and decrease the TTL by 1. If the TTL value is not zero and there is no number, then the protocol sets a schedule to forward the message. The packet will be delayed for a random period between T_{min} and T_{max} , and then relayed to its neighbors that are one or more hops away from the source. In this specific reference case, our approach simply rebroadcasts the packet. A packet propagates outward from the source node up to TTL hops away in this mode. The random delay before forwarding a message is necessary to avoid collisions because ready-to-send/clear-to-send (RTS/CTS) dialogues are inappropriate in broadcasting operations when the timing of rebroadcasts among interfering nodes can be highly correlated.

T_{min} has several considerations. First, there is a need to provide a time-buffer for local packet recovery. One of the main motivations behind our paradigm is to recover lost packets quickly among immediate neighboring nodes within a controllable time frame. T_{min} associated with the relay operation provides an opportunity for a node to hear the same message from other rebroadcasting nodes before it would actually have started to transmit the message. A counter is used to keep track of the number of times the same broadcast message is heard. If the counter reaches 4 before the scheduled rebroadcast of a message, then the transmission is cancelled and the node will not relay the specific message

3. 3 Relay Initiated Error Recovery

Once a sequence number gap in a file's fragments is detected, a node goes to relay initiated error recovery is the proactive act of requesting a retransmission from neighboring nodes once loss is detected at a receiving node. Our protocol uses the concept of "loss aggregation" whenever loss is detected, that is, it attempts to batch up all message losses in a single error recovery operation whenever possible.

3.3.1 Loss Aggregation:

Data loss is often correlated in time because of fading conditions and other channel impairments. As a result, loss usually occurs in batches (bursty loss. Our protocol aggregates loss such that the fetch operation deals with a "window" of lost packets instead of a single-packet loss. In a dense network where a node usually has more than one neighbor, it is possible that each of its neighbors only obtains or retains part of the missing segments in the loss window.

Our approach allows different segments of the loss window to be recovered from different neighbors. In order to reduce redundant retransmissions of the same segment, each neighbor waits for a random time before transmitting segments. Other nodes that have the data and scheduled retransmissions will cancel their timers if they hear the same "repair" from a neighboring node. In poor radio environments, successive loss could occur including loss of retransmissions and fetch control messages. Therefore, it is not unusual to have multiple gaps in the sequence number of messages received by a node after several such failures. Aggregating multiple loss windows in the fetch operation increases the likelihood of successful recovery in the sense that as long as one fetch control message is heard by one neighbor, then all the missing segments could be resent by this neighbor.

3.3.2 Error Recovery Timer:

In relay initiated recovery mode, a node aggressively sends out negative acknowledgment (NACK) messages to its immediate neighbors to request missing segments. If no reply is heard or only a partial set of missing segments are recovered within a period (this timer defines the ratio between relay and recovery, as discussed earlier), then the node will resend the NACK every interval . To avoid the message implosion problem NACK messages never propagate; that is, neighbors do not relay NACK messages unless the number of times the same NACK is heard exceeds a predefined threshold, while the missing segments requested by the NACK message are no longer retained in a node's data cache. In this case, the NACK is relayed once, which in effect broadens the NACK scope to one more hop to increase the chances of recovery. Each neighbor that receives a NACK message checks the loss window field. If the missing segment is found in its data cache, the

neighboring node schedules a reply event. Neighbors will cancel this event whenever a reply to the same NACK for the same segment is overheard. In the case where the loss window in a NACK message contains more than one segment to be resent or more than one loss window exists in the NACK message

3.3.3 Proactive Recovery

As in many NACK systems, the fetch operation described previously is a reactive loss recovery scheme in the sense that a loss is detected only when a packet with a higher sequence number is received. This could cause problems on rare occasions; for example, if the last segment of a file is lost there is no way for the receiving node to detect this loss since no packet with a higher sequence number will be sent. In addition, if the file to be injected into the network is small (e.g., a script instead of binary code), it is not unusual to lose all subsequent segments up to the last segment following bursty loss. In this case, the loss is also undetectable and, thus, nonrecoverable with such a reactive loss detection scheme. In order to cope with these problems, our approach supports a timer-based "proactive fetch" operation such that a node can also enter the fetch mode proactively and send a NACK message for the next segment or the remaining segments if the last segment has not been received and no new packet is delivered after a period of time T_{pro}. The proactive fetch mechanism is designed to automatically trigger the fetch mode at the proper time. If the fetch mode is triggered too early, then the extra control messaging might be wasted since upstream nodes may still be relaying messages or they may not have received the necessary segments. In contrast, if the fetch mode is triggered too late, then the target node might waste too much time waiting for the last segment of a file, significantly increasing the overall delivery latency of a file transfer.

Assume that the packet loss rate P stays constant during the controllable time frame, it can be shown that in a negative acknowledgment system, the probability of a successful delivery of a packet between two nodes that allows K retransmissions can be expressed recursively as

$$(1-p)+p\times\Omega(k) \qquad (k\geq 1)$$
$$\Omega(k)=\Phi(1)+\Phi(2)+\cdots\Phi(k)$$
$$\Phi(k)=(1-p)^2 \times[1-p-\Phi(1)-\cdots-\Phi(k-1)] \qquad (\Phi(0)=0).$$

Where $\Omega(k)$ is the probability of a successful recovery of a missing segment within k retransmission (k) is the probability of a successful recovery of the missing segment at kth retransmission. The above expressions are evaluated numerically against the packet loss rate, as shown in Fig. 1, demonstrating the impact of increasing the number of retransmissions up to equal to 7. We can see that substantial improvements in the success rate can be gained in the region where the channel error rate is between 0% and 60%. However, the additional benefit of allowing more retransmissions diminishes quickly and becomes negligible when is larger than 5. This simple analysis implies that the optimal ratio between the timers associated with the relay and error recovery operations are approximately 5.

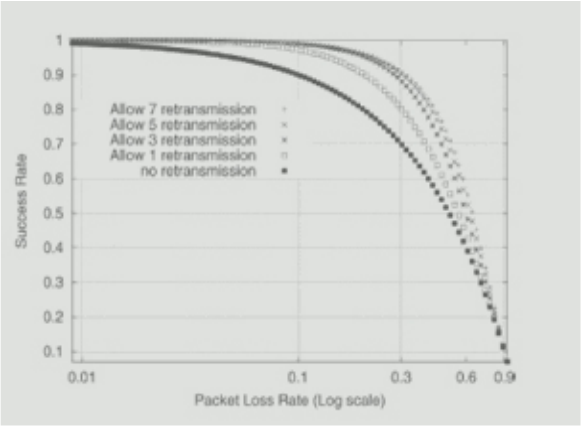

Fig.1. Probability of successful delivery of a message over one hop when the mechanism allows multiple retransmissions before the next packet arrival.

3.4 Selective Status Reporting

Our protocol supports a selective report operation designed specifically to feedback data delivery status to users in a simple and scalable manner. In wireless communication, it is well known that the communication cost of sending a long message is less than sending the same amount of data using many shorter messages [5]. Given the potential large number of target nodes in a sensor network in addition to potential long paths (i.e., longer paths through multihops greatly increases the delivery cost of data), the network would become overwhelmed if each node sent feedback in the form of report messages. Therefore, there is a need to minimize the number of messages used for feedback purposes. Anode enters the report mode when it receives a data message with the "report bit" set in the message header.

The user node sets the report bit in its injected message whenever it needs to know the latest status of the surrounding nodes. To reduce the number of report messages and to avoid report implosion only the last hop nodes, (i.e., TTL=1) will respond immediately by initiating a report message by sending it to its parent node, where the previous segment came from, at a random time between (0, Δ).

Each node along the path towards the source node will piggyback their report message by adding their own status information into the report, and then propagate the aggregated report toward the user node. Each node will ignore the report if it found its own ID in the report to avoid looping. Nodes that are not last hop nodes but are in report mode will wait for a period of time (T_{report}= T_{max}*TTL+ Δ) sufficient to receive a report message from a last hop node, enabling it to piggyback its state information. A node that has not received a report message in the report mode will initiate its own report message and send it to its parent node. If the network is very large, then it is possible for a node to receive a report message that has no space to append more state information. In this case, a node will create a new report message and send it prior to relaying the previously received report that had no space remaining to piggyback its state information. This ensures that other nodes en-route toward the user node will use the newer report message rather than creating new reports because they themselves received the original report with no space for piggybacking additional status.

3.4.2 Single-Packet Message Delivery

There is need to support the reliable deliver of single-shot atomic messages in sensor networks, for example, in support of reliable control and management of sensors. For messages that fit into a single packet (e.g., smaller than the network MTU), delivery failure is undetectable using NACK-based protocol without the addition of explicit signaling. This is because our protocol detects loss by observing sequence number gaps or timeouts. To address this service need, our approach makes use of its reporting primitive to acquire application-specific feedback at the sink. This protocol sets the report bit at the sink in every single-packet message that requires reliable delivery. Based on the feedback status, the sink resends the packet until all receivers confirm reception. This essentially turns the protocol into a positive aggregated- ACK protocol used in an on-demand manner by the sink for these special case messages. The use of the report mechanism to support reliable data delivery of single-shot atomic messages highlights the flexible use of our protocol mechanisms to meet application specific needs.

3.5 Node Supervising

To localize the topology changes due to mobility we use some aspects of SPIN protocol [Sensor Protocols for Information via Negotiation] [4].We introduces the concept of neighbor list because each node need know only its single hop neighbors. In mobile conditions, immediate neighbors are identified using neighbor list and the packets are broadcasted. Changes in the local topology can trigger updates to a node's neighbor list. If a node notices that its neighbor list has changed, it can spontaneously re-send its data to new neighbors. Hence it is an efficient method to cope with topology change. We can also use the neighbor list for retrieving a lost packet by sending NACK messages aggressively to immediate neighbors.

4. Performance Evaluation

In order to highlight the different design choices made, we compare the performance of our approach with PSFQ and SPIN. The major purpose of our comparison is to highlight the impact of different design choices made. PSFQ is a proven reliable transport protocol, but lacks mobility. So our protocol stands taller than PSFQ. We choose three metrics that underpin the major motivation behind the design of our protocol and compare it with PSFQ

• *Average Delivery Ratio*, which measures the ratio of the number of messages a target node received to the number of messages a user node injects into the network. This metric indicates the error tolerance of a scheme at the point where a scheme fails to deliver 100% of the messages injected by a user node within certain time limits. The result (Fig.2) shows a slight increase in the Average Delivery Ratio than the existing PSFQ

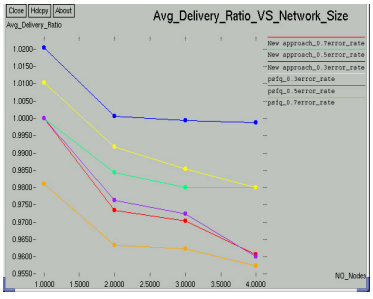

Fig.2.Comparison of average delivery ratio as a function of network size

• *Average Delivery Overhead*, which measures the total number of messages sent per data message received by a target node. This metric examines the communication cost to achieve reliable delivery over the network. The result (Fig.3) shows slight decrease in the average Delivery Overhead than PSFQ.

• *Average Latency*, which measures the average time elapsed between the transmissions of the first data packet from the user node until the reception of the last packet by the last target node in the sensor network. This metric examines the delay bound performance of a scheme. The result (Fig.4) shows a slight decrease in the Average Delivery Latency than PSFQ.

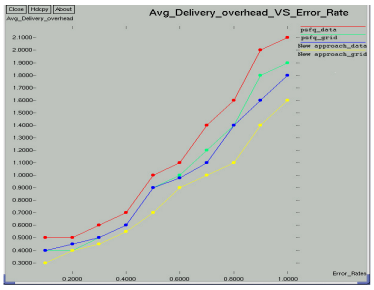

Fig.3 Comparison of average delivery ratio and error rate

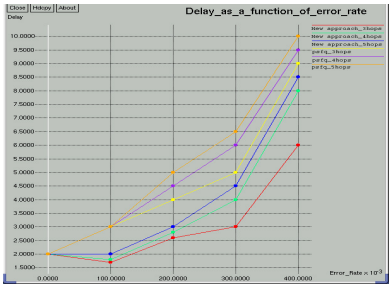

Fig.4. Comparison of Average Latency as a function of error rate

Then we compare our protocol with SPIN. The SPIN includes a family of protocols. They incorporate negotiation before transmitting data in order to ensure that only useful information will be transferred. The reason for comparing our approach and SPIN is that these protocols are well suited for environment where the sensors are mobile because they base their forwarding decisions on local neighborhood information. In both of these protocols, each node only know its single-hop neighbors, the topology change is easily localized. SPIN cannot guarantee delivery of data. Assume that nodes interested in data are located far away from the source node, and the nodes between source and destination are not interested in data, such data will not be delivered to the destination at all. But our approach is well suited for reliable transport in mobility conditions also.

5. Related Work

Here we contrast some of the contributions [7] & [8] for reliable data delivery in sensor networks. Reliable multisegment transport (RMST) [7] is a transport layer paradigm for sensor networks that is closest to our work. RMST is designed to complement directed diffusion [1] by adding a reliable data transport service on top of it. RMST is a NACK-based protocol like our approach, which has primarily timer driven loss detection and repair mechanisms. Unlike our approach which provides reliability purely at the transport layer, RMST involves both the transport and MAC layers to provide reliable delivery.

In ESRT [8], the authors propose using an event-to-sink reliability model in providing reliable event detection that embeds a congestion control component. In contrast to our approach, ESRT does not deal with data flows that require strict delivery guarantees; rather, the authors define the "desired event reliability" as the number of data packets required for reliable event detection that is determined by the application. But all of the above contributions do not support mobility.

6. Conclusion

In this paper we have proposed a reliability transport protocol for wireless sensor networks that can be used in various error as well as mobility conditions. Our proposed methodology can outperform existing protocols and is highly responsive to the various error and mobility conditions experienced in sensor networks. Nevertheless, there remain a number of open questions regarding the overhead caused by triggering updates in neighbor list. If there is high mobility, the overhead is high. The future work is to minimize this overhead caused by high mobility conditions

References

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. 6th Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 2000, pp. 56–67.

[2] C-Y.Wan, A.T.Campbell, and L.Krishnamurthy, "PSFQ: A reliable transport protocol for wireless sensor networks," in *Proc. 1st ACM Int.Workshop Wireless Sensor Netw. Appl. (WSNA)*, Atlanta, GA, Sep. 28, 2002, pp. 1–11.
[3]W. Heinzelman ,A. Chandrakasan, and H.Balakrishnan," Energy-Efficient Communication Protocol for Wireless Micro sensor Networks," *Proc.33rd Hawaii Int'l. Conf. Sys. Sci*., Jan 2000
[4] C. E. Perkins and P. Bhagwat, "Highly dynamic destination- sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. Symp. Commun. Arch. Protocols (SIGCOMM)*, Sep. 1994, pp. 212–225.
[5] D. A. Maltz, "On-demand routing in multihop wireless mobile ad hoc networks," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, 2001.
[6] S. Floyd, V. Jacobson, C. Liu, S. Macanne, and L. Zhang, "A reliable multicast framework for lightweight session and application layer framing," *IEEE/ACM Trans. Netw.*, vol. 5, no. 2, pp. 784–803, Dec. 1997.
[7] R. Stann and J. Heidemann, "RMST: Reliable data transport in sensor networks," in *Proc. 1st IEEE Int. Workshop Sensor Net Protocols Appl. (SNPA)*, Anchorage, AK, May 2003, pp. 102–112.
[8] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz, "ESRT: Event-to-sink reliable transport in wireless sensor networks," presented at the ACM MobiHoc, Annapolis, MD, Jun.

Bejoy.B.J He completed his B.Tech in I.T and M.E in CSE under Anna University, Chennai. He is working as an Assistant Professor in CSE Department of P.S.R Engineering College, Sivakasi, Tamilnadu. He has 4 years of teaching experience

Dr.B.Paramasivan. He completed his PhD from Anna University Chennai in the year 2009 in the area of Wireless Sensor Networks. He is the Professor and Head of CSE department, National Engineering College, Kovilpatti, Tamilnadu. He has published many papers in reputed journals. He has a total of 23 years of working experience.