A Hop-by-hop Reliability Support Scheme for Wireless Sensor Networks

Haejun Lee, Yangwoo Ko, Dongman Lee
Information and Communications University
School of Engineering
Daejeon, Korea
{haejun, newcat, dlee}@icu.ac.kr

Abstract

The most significant obstacle to multi-hop communications in wireless sensor networks is high link error rate. Thus, an efficient hop-by-hop reliability support scheme is highly required. In this paper, we identify the characteristics of two typical communication patterns in wireless sensor networks and address the problems of previous end-to-end sequence based hop-by-hop error recovery protocols, which cannot work properly with route change events and have a scalability problem with multiple senders. We propose a new hop-by-hop reliability support scheme named HRS. It uses hop-by-hop sequence numbers for hop-by-hop error recovery and operates in two separate modes depending on communication patterns. Through ns-2 based simulations, we show that both modes of HRS outperform the existing protocols in most metrics.

1. Introduction

Several transport layer hop-by-hop error recovery protocols have been proposed for wireless sensor networks (or WSNs) [1, 2]. Most of them argued that providing hop-by-hop reliability in a transport-layer rather than in a MAC layer is more efficient since more sophisticated approaches are applicable. In these protocols, intermediate nodes cache and recover packets to increase delivery ratio of the links and consequently deliver data more efficiently.

In WSNs, there are many factors that may cause route changes in the middle of a session. Failure of a sensor node constituting a route incurs a route change, which invites one or more new nodes. These new nodes are not necessarily receivers of the session since only subsets of nodes can be receivers even in "a sink to sensors" communications [3]. In previous approaches, there is no way to tell the starting sequence of a session to newly joined nodes when NACK based error recovery is used. Thus, they cannot work properly in the presence of route changes. Especially, in "sensors to a sink" communications, protocols for which intermediate nodes are required to maintain sessions for each sender as is done in the previous approaches will not scale well since all sensors are senders in this communication pattern.

In this paper, we propose a new hop-by-hop reliability support scheme called HRS. A hop-by-hop sequence number represents an actual sequence of packets between a node and its next hop node(s). This number is devised for hop-by-hop error detection and recovery to perform robustly in the presence of route changes, detect packet losses in the hop where the packets are lost, and reduce session management overhead when there are multiple senders. HRS operates in two separate modes depending on communication patterns; (1) *unicast mode* for "sensors to a sink" communications and (2) *broadcast mode* for "a sink to sensors" communications.

We evaluate both modes of HRS with two typical application scenarios in the ns-2 simulator [4]. The results show that HRS outperforms the compared protocols in most metrics and works robustly and consistently with route changes.

The rest of this paper is organized as follows. Section 2 describes related works and section 3 presents design considerations and the details of the HRS. The evaluation results are presented in section 4 and we conclude our work in section 5.

2. Related works

Several protocols have been proposed for hop-by-hop error recovery in transport layer for WSNs. They are appropriate when link error rate is high [1, 2]. ESRT proposed a reliable event detection concept. [5]

PSFQ [1] is a well known reliable data transport protocol designed for code distribution. It proposed a "pump slowly fetch quickly" scheme for hop-by-hop error recovery to solve the "propagation of a NACK event" problem [1]. PSFQ forwards data packets at a relatively slow speed. If a packet is lost, it suppresses forwarding of next packets until the lost packet is recovered to prevent the problem. However, suppression of next packets results in longer completion time and bigger buffer space requirements since packet forwarding

Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)
0-7695-2520-2/06 $20.00 © 2006 IEEE

is suppressed while following packets do not stop coming. PSFQ uses a sequence number of an end-to-end session for error recovery. Every intermediate node should maintain per-sender sequence numbers. Thus, it will suffer from high session management overhead especially in "sensors to a sink" communications. Also, it does not work properly if end-to-end session information in intermediate nodes is unavailable due to route changes, which may happen frequently in WSNs.

ESRT [5] proposed an event-to-sink reliable transport protocol. They assumed that reliability level of event detection is directly related to the number of packets received in a period. A sink controls reporting rates of sensors to get desired reliability level based on the number of packets received in the previous period.

3. HRS

In this section, we discuss design considerations and describe the proposed scheme, Hop-by-hop Reliability Support (HRS). Two novel features of HRS are (1) hop-by-hop sequence number based error recovery with two modes and (2) a hybrid of hNACK and delayed hACK.

3.1. Terminologies

- *End-to-end sequence number,* which is assigned to each data packet by a sender to represent the order of the packet within the whole sequence of data packets sent by the sender to a receiver (or a set of receivers) for the current session.
- *Hop-by-hop sequence number*, which is assigned to each data packet by a node to represent actual sequence of forwarded packets between the node and its next hop node(s) toward the destination.
- *Hop-by-hop ACK, NACK, and Repair (hACK, hNACK, and hRepair)*, which are control packets for hop-by-hop error recovery. We put 'h' in front of names for distinction from end-to-end session control packets.
- *Hop-by-hop ACK reply timer,* which is used to delay sending hACK packets and is canceled if there is another packet following the current one before expiration.

3.2. Design considerations

- *Use of different reliability support depending on communication patterns* - Two typical communication patterns in WSNs are *sensors to a sink* for event detection and *a sink to sensors* for sensor retasking. In the former pattern, sensors send packets via unicast since only one node on the route toward a sink is to receive the packets. The other significant point is that all sensors are senders in this pattern. Protocols like PSFQ that make intermediate nodes maintain sessions such as per-sender sequence numbers will suffer from scalability due to high session management overhead in intermediate nodes. In the latter pattern, sensors use broadcast rather than unicast to send packets to multiple next hop nodes at once. Since there is only one sender, a sink, maintaining sessions in intermediate nodes does not affect scalability significantly nor incur high session management overhead. A hop-by-hop error recovery should consider these characteristics of each communication pattern.
- *Efficient reliability support in the presence of route changes* - Failures of nodes on the route result in route changes. A hop-by-hop error recovery scheme should perform robustly in the presence of route changes. If end-to-end sequence numbers are used for hop-by-hop error recovery as done in the previous works and a new node joins in the middle of the session, the new node is unaware of the sequence number of the most recently received packet. Thus, the node cannot immediately participate in error recovery. To overcome this, nodes should be able to perform error recovery even when they lately joined the route in the middle of the session.
- *Recover an error where it occurs* - A gap in the end-to-end sequence number does not necessarily mean that packets are lost in the hop between the node and its predecessor. The discrepancy between the sequence of packets observed in an end-to-end session and the one observed in a hop is the source of the "propagation of a NACK event" problem [1]. If packet losses are recovered in the hop where the packets are lost, there can be no such discrepancy.
- *Less control overhead* – For the sake of efficiency, NACK based approaches are more appropriate for recovering from losses. However, NACK based approaches suffer from typical problems such as a loss of whole messages and that of last few packets. Thus, hop-by-hop error recovery should combine the efficiency of NACK based approaches while avoiding loss of whole message or last few packets problems.

3.3. Hop-by-hop sequence numbering

In HRS, data packets are newly sequenced by a hop-by-hop sequence number in the order of departure from a node to its next hop node(s) on a route. Since packets are newly sequenced in every intermediate node, a node can detect losses of packets that are actually lost in the hop between that node and its previous node. Depending on communication patterns, HRS works in two separate modes.

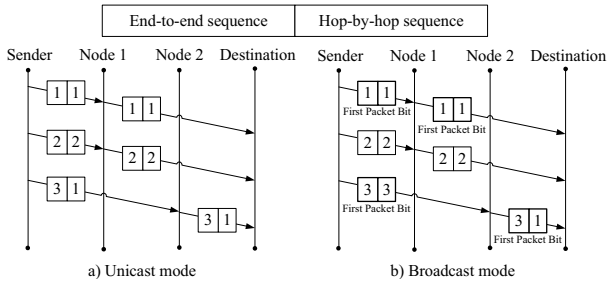

Fig. 1. Forwarding using HRS with a route change event

3.3.1. Unicast mode. This mode is for "sensors to a sink" communications. In this mode, hop-by-hop sequence numbers are used between a node and each of its next hop nodes independently. Every node maintains a hop-by-hop sequence number for each neighbor node and intializes it to zero when it sends a packet to that node for the first time. This sequence number is managed continuously regardless of the sessions that the node involves. Since hop-by-hop sequence numbers are only meaningful between a node and one of its next hop nodes regardless of sessions, intermediate nodes do not have to maintain any information about sessions and senders. This promises scalability in sensors to a sink communications.

By managing hop-by-hop sequence numbers regardless of sessions, newly joined nodes do not need to know anything about the session. Thus, they can immediately join a route. Fig. 1 a) shows how the unicast mode of HRS deals with a route change. The first two packets are delivered to the destination via node 1 and the third packet is delivered via node 2. With an end-to-end sequence number based approach, node 2 regards two previous packets as being lost when it receives the third packet since the sequence number of the packet is 3.[1] So node 2 requests for recovery of first two packets, which have been delivered to the destination already. In the unicast mode of HRS, when node 2 receives the third packet from a sender, it can figure out that there was no loss before the packet since the hop-by-hop sequence number of the packet is 1.

One of advantages of this mode is that packets from different senders are sequenced by the same hop-by-hop sequence number if they pass the same links. Thus, the intermediate nodes can have consecutive sequences of packets from different senders by bundling these packets even if each of the senders sends only a single packet. This makes NACK based error recovery more feasible in sensors to a sink communications.

3.3.2. Broadcast mode. This mode is for "a sink to sensors" communications. The difference of this mode from the unicast mode is that the intermediates do not maintain individual hop-by-hop sequence numbers for each next hop node, but only one for all the next hop nodes for a session. Every node in a session maintains a hop-by-hop sequence number for all the next hop nodes and initializes it to zero when a session is created. This is to broadcast packets to the next hop node of the session at once with one hop-by-hop sequence number.

In this mode, however, a newly joined node needs to know the starting hop-by-hop sequence number of the forwarder. Otherwise, it cannot immediately join a session. To allow an immediate join, the "*first packet bit*" that marks the first packet toward the next hop nodes that just joined is added to the packet header. Whenever a set of the next hop nodes changes, the previous node sends a packet with the "first packet bit" enabled. If a newly joined node receives the first packet with the "first packet bit" enabled, it knows that there were no losses before this packet. If the first received packet does not have the bit enabled, it knows that some packets before this packet are lost. Fig.1 b) shows how the broadcast mode of HRS performs in the presence of a route change. Unlike the unicast mode, a hop-by-hop sequence number of the third packet from a sender to a newly joined node, i.e. node 2, is 3 and packets that are delivered to the next hop node for the first time has the "first packet bit" enabled. Even the first packet from a sender to node 2 has hop-by-hop sequence number of 3, node 2 can know that there were no losses before this packet by the "first packet bit". If it does not have the "first packet bit" enabled, it can recover the lost packets by sending an hNACK message for the previous packets from the actual first packet.

3.4. Hybrid of hNACK and delayed hACK

To avoid the typical problems of NACK based approaches such as losing a whole message or last few packets, at least the last packet comprising a message should be delivered reliably. For this, HRS uses a NACK-based approach while applying an ACK based approach to the last packet. However, intermediate nodes do not know which packets comprise a message since HRS does not refer to any end-to-end sequence numbers. Thus, an intermediate node in HRS regards any packet that is not followed by a next packet within a reasonable time period as the last packet. This is realized by a pair of timers described below.

When sending a data packet, a node sets an hACK

[1] Cost of route changes may be reduced if newly joined nodes recover losses only from the lowest sequence number of received packets and leave recovery of first few packets for next nodes who have maintained end-to-end session information as in RMST. However, multi-hop recovery of those packets will result in NACK explosion and the most significant side effect is that it will happen every time when starting the session even there is no route change.

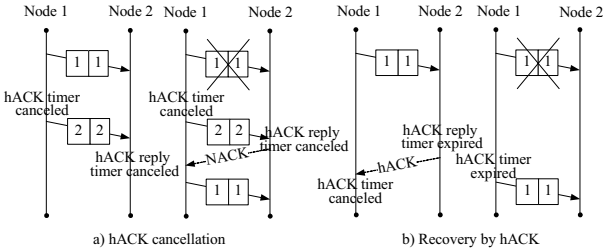

Fig. 2. Delayed hACK approach of HRS

timer and another node receiving the packet establishes an hACK reply timer. Fig. 2 a) shows how hACK packets are suppressed if there is a next data packet following the same hop before the timers expire. Since both timers are canceled by the next packet, an hACK packet is not sent. If the previous packet is lost, it can be detected by the next data packet as shown in the right figure of Fig. 2 a).

Losses of the last packets that are not followed by other packets are detected using the hACK approach within a reasonable time. Fig. 2 b) shows the recovery of the last packet. A receiver sends an hACK message when the hACK reply timer expires and a sender cancels the hACK timer. If the previous packet is not delivered successfully or an hACK message is lost, a sender retransmits the packet when the hACK timer expires.

When sending multiple packets, most of hACK packets will be suppressed by the following packets and their losses can be detected by a gap in hop-by-hop sequence numbers except last few packets. Last few packets will be delivered reliably by taking advantage of hACK packets.

Longer timeout values of the hACK reply timer will delay recovery of losses longer, but requires less hACK messages. Shorter timeout values will recover losses earlier even there is no next packet by consuming more hACK messages. Thus, by varying the timeout value of the hACK reply timer, applications can trade-off between control overhead and timeliness of packet delivery.

4. Evaluations

In this section, we evaluate two modes of HRS using scenarios of two typical communication patterns with ns-2 simulator [4]. The unicast mode is evaluated using a scenario of ESRT [5] and compared with no hop-by-hop reliability and pure-ACK based approach. The broadcast mode is compared with PSFQ using a simple code distribution scenario with a route change event manipulation. IEEE 801.11 is used as a MAC protocol. The transmission range of each node is 250m and the distance between sensors is 200m. In order to examine schemes with various error rate settings, we add a uniform error to a MAC layer.

4.1. Unicast mode

The unicast mode of HRS is compared with a pure-ACK approach and without hop-by-hop error recovery mechanism. HRS and pure-ACK approach are used to support hop-by-hop reliability for ESRT. Nodes are deployed in a 5x5 grid and a sink is the node in the center. A period for reporting rate control is 10 seconds and target level of reliability is 100 packets per a period. Size of data packets is 64 bytes. Timeout values of the hACK reply timer of HRS are 1 and 10 seconds. Simulation ran for 900 seconds. We choose two metrics to justify the advantages of the unicast mode of HRS.

- *Energy Efficiency*, which is the number of bytes transmitted for delivering one data packet to a sink. Since packet sizes used by each protocol are different, results are normalized by the packet sizes.
- *Uniformity of results,* which is standard deviation of the number of packets received from each sensor. It shows how fairly results are distributed to each sensor.

Fig. 3 shows the average number of bytes transmitted for delivering one data packet to a sink. HRS with 10 seconds of the hACK reply timer (or HRS-10) shows the best efficiency since more losses are detected by the NACK based approach, which uses less control packets. In HRS with 1 second of the hACK reply timer (or HRS-1), more losses are recovered by hACK messages than in HRS-10 because of a shorter timeout value. However, HRS-1 can recover losses earlier. The pure-ACK approach and no hop-by-hop reliability reveal similar efficiency. This is because the pure-ACK approach uses a lot of control packets while no hop-by-hop reliability achieves a required level of reliability by higher sending rate of sensors.

Fig. 4 represents the uniformity of packet deliveries by standard deviation of the number of packets from each sensor. When there is no hop-by-hop reliability support, the standard deviation is relatively high even if link error rate is low. In this case, packets from far sensors have a lower probability of successful delivery to the sink than packets from near sensors. With hop-by-hop reliability support like other cases, standard deviations are almost zero since almost all packets from each sensor arrive without packet losses in the intermediate nodes.

4.2. Broadcast mode

We compare the broadcast mode of HRS with PSFQ. A simple topology of 6 nodes in a line is used to see an effect of a route change. A route change is manipulated to happen once during a session by replacing three nodes with new nodes. Route change timing is randomly

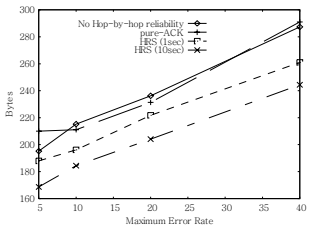

Fig. 3. Bytes of transmission for delivering one data packet

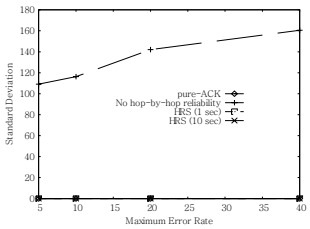

Fig. 4. Standard deviation of numbers of packets from each sensor

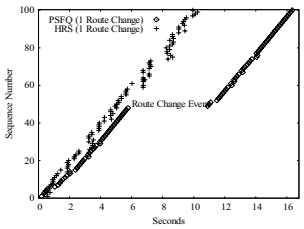

Fig. 5. Packet sequence traces with a route change

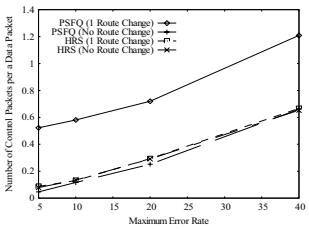

Fig. 6. Numbers of control packets per one data packet

selected and all simulation results (except packet sequence traces) are average values of 10 simulation runs. The sending rate at all the intermediate nodes in PSFQ and that of the sender in HRS are 10 packets per second. In total, a hundred 50-byte packets are sent using MAC broadcast without any reliability support at a MAC layer. We choose two metrics to justify the advantages of the broadcast mode of HRS.

- *Packet sequence traces*, which shows the throughput and delivery sequence of each approach.
- *Control overhead*, which shows the number of control packets required for transferring one data packet to the next hop.

Fig. 5 shows packet sequence traces with a route change when sending the 51st packet. In case of PSFQ, there clearly exists a blackout period while HRS shows no significant changes. In PSFQ, a newly joining node on the route has to repair packets from the first packet to the 50th. The node cannot forward any next packets before recovery is done. HRS performs slightly faster than PSFQ even when there is no route change since HRS does not stop forwarding to recover previous packets. Another difference is that more packets are delivered in out of order in HRS since in-sequence delivery is not forced at the intermediate nodes. However, PSFQ does not provide in-sequence delivery in end-to-end manner, either.

Fig. 6 shows the control overhead. Without a route change, PSFQ shows scarcely lower control overhead than HRS when an error rate is low since HRS uses hACK messages. PSFQ uses the "proactive NACK" approach to solve the last few packet problem [1], in which a receiver sends a NACK packet proactively if a next packet does not arrive within a reasonable time.[2] As error rate increases, the difference between PSFQ and HRS decreases because PSFQ needs to use more "proactive NACK" packets. With a route change, the control overhead of PSFQ is significantly higher since it has to recover all the packets sent before the route change. HRS has almost no difference regardless of route changes.

[2] However, PSFQ cannot detect a loss of the whole message while HRS does.

5. Conclusion

In this paper, we have identified the characteristics of two typical communication patterns in WSNs and address the problems of previous works. From these observations, we proposed HRS, a hop-by-hop sequence numbering based hop-by-hop reliability support scheme to overcome high link error rate in WSNs.

By using hop-by-hop sequence numbers instead of end-to-end sequence number, each intermediate node is able to detect actual packets that are lost between itself and its previous node. HRS achieves scalability and efficiency with multiple senders or with multiple receivers by operating in two modes depending on communication patterns. By delaying and suppressing hop-by-hop acknowledgement messages, it can reliably deliver single packets as well as consecutive packets while enhancing efficiency of NACK based approach.

Through ns-2 based simulations, we showed that both modes of HRS perform better than the existing protocols in terms of efficiency, fairness of results, and completion time.

HRS is going to be implemented in an actual sensor network test-bed and issues taken from real world experiments will be applied for designing a reliable data transport framework for WSNs.

6. References

[1] C.-Y. Wan, A. T. Campbell, *PSFQ: A reliable transport protocol for wireless sensor networks*, in proc. of ACM WSNA, 2002.

[2] F. Stann and J. Heidemann, *RMST: Reliable data transport in sensor networks*, in proc. of IEEE SNPA, 2003.

[3] E. Welsh, W. Fish, P. Frantz, GNOMES: *A Testbed for Low-Power Heterogeneous Wireless Sensor Networks,* in proc. Of IEEE ISCAS, 2003

[4] *The Network Simulator - ns-2*. http://www.isi.edu/nsnam/

[5] Yogesh Sankarasubramaniam, Ozgur B. Akan, and Ian F. Akyildiz. *ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks,* in proc. of ACM MobiHoc, 2003.