iACK: Implicit Acknowledgements to Improve Multicast Reliability in Wireless Sensor Networks

Kittithorn Tharatipayakul*, Steven Gordon*, Kamol Kaemarungsi†
*School of Information, Computer and Communication Technology,
Sirindhorn International Institute of Technology, Thammasat University, Thailand
Email: kittithornarm@gmail.com, steve@siit.tu.ac.th
†National Electronics and Computer Technology Center (NECTEC)
National Science and Technology Development Agency (NSTDA), Thailand
Email: kamol.kaemarungsi@nectec.or.th

***Abstract*—Reliability in IP-multicast forwarding becomes useful while operating in lossy environment of wireless sensor networks. RPL is a protocol for forming a multicast tree in sensor networks for multicast data forwarding. Trickle multicast uses RPL to provide reliability, however it has a high overhead and delay, especially when a radio duty cycle is used on the sensors. Stateless Multicast Forwarding (SMRF) optimizes for the radio duty cycle, reducing delay but also reducing reliability. This paper proposes iACK, a retransmission scheme on top of SMRF that uses implicit acknowledgements present in wireless broadcast to determine which packets a node should retransmit. We have implemented iACK in ContikiOS. Results show that iACK delay is about 5 times less than Trickle (and close to SMRF), and packet delivery ratio is about 80% (compared to between 20% and 60% for Trickle and SMRF). With a slight increase in memory requirements, iACK offers a valuable trade-off compared to existing protocols.**

Keyword: Reliable Multicast, Trickle, Contiki, Sensor Networks

I. Introduction

A characteristic of communication protocols in WSNs is that they often have to support one-to-many communications (e.g. a sink node distributes a command to many sensor nodes). Unacknowledged (and/or unreliable) multicast delivery is usually a common mechanism for WSNs [1] [2] to support one-to-many communications. However, there are some applications that require *reliable* data delivery when using mutlicast in WSNs. An example is a file distribution, such as over-the-air (OTA) updating of firmware image, from a sink node to many sensor nodes. This paper presents a technique for providing reliable multicast data delivery in WSNs.

In communication network protocols, reliability is often achieved with either automatic repeat request (ARQ) mechanism, i.e. retransmission, or with forward error correction (FEC) mechanism, or a combination of the two mechanisms. Reliability may be provided by one or more protocols in a layered protocol stack. Link layer protocols, such as IEEE 802.15.4 [3], often use ARQ across a single wireless hop to overcome packet loss due to received signal degradation (e.g. fading effect and interference from other sources). However, even with link layer retransmissions, end-to-end reliability across multiple wireless hops is not guaranteed. Packets may still be lost due to buffer overflow at intermediate and end nodes or due to unacknowledged broadcasting both in link and network layers. In WSNs, such packet losses can become significant performance degradation when combined with the contentions among nodes and the limited transmission opportunities of nodes due to sleeping cycle. Note that end-to-end reliability is often provided by transport (or application) protocols such as transport control protocol (TCP) but such protocols are usually not deployed in WSNs.

End-to-end retransmission offered by TCP and/or other protocols is not well-suited to multicast delivery because of the storm of explicit acknowledgement (eACK) packets that are returned from many nodes to a single node. Therefore researchers have used the behavior of wireless broadcast to act as implicit acknowledgement (iACK) [4]. Another alternative is FEC; however, it can add significant complexity on already resource-limited sensor nodes. Therefore, researchers are interested in developing specific mechanisms for WSNs that work with network routing protocols to aid in multicast data delivery. Two key efforts are Trickle [5] and SMRF [6]. Trickle is an algorithm used to schedule (re-)transmissions on a hop-by-hop basis to increase reliability but can create long communication delay when multicasting through a WSN. SMRF takes the RPL built-in information and cross-layer design approach to decrease congestion and makes packet delivery ratio become higher. In this paper, we introduce iACK, which builds on features of SMRF but adds an implicit acknowledgement scheme to increase reliability while maintaining a low overhead.

The rest of this paper is organized as follows. Section II explains the network protocol and routing mechanisms assumed in our WSNs and gives more details about Trickle and SMRF. Section III presents our proposed iACK. Section IV reports on our simulation results comparing iACK with other existing approaches. Section V concludes the paper.

II. Background and Literature Review

To interconnect WSNs with existing networks, IPv6 is recommended as a common networking protocol. *IPv6 over Low power WPAN (6LoWPAN)* [7] is an effort by Internet Engineering Task Force (IETF) to support IPv6 on WSNs. We assume routing in 6LowPAN is used in this paper. We assume that the IETF developed protocol *IPv6 Routing Protocol for Low-Power and Lossy Networks* (RPL) [1] is used. Section II-A explains concepts of RPL, while Sections II-B and II-C describes the special case of multicast routing, the proposed improvements considered in this paper, SMRF and Trickle.

A. Routing and Forwarding with RPL

Due to the low storage and processing resources of sensor nodes, normal IP routing protocols are not recommended for WSNs. RPL is suggested as a refined routing mechanism that uses selective flooding in the network. It is designed for the case when there is one source (e.g. gateway between WSN and external network, called a LOWPAN Border Router (LBR)) and multiple destinations (sensors) and relies on building a destination-oriented directed acyclic graph (DODAG).

RPL operates in four stages. The *initial build* stage involves the LBR performing a network-wide broadcast (i.e. full flooding) of a DODAG Information Object (DIO) message. This message contains information about the LBR as well as a rank for nodes. LBR has rank 0 while other nodes have higher ranks. The default approach is that the rank increments for each hop away from the LBR. Next stage, called *Purpose assignment*, is a rule used to calculate rank (i.e. lower rank than x, battery above y%). Nodes that can meet these requirements set their rank to be low numbers. When the next DIO message is sent, lowest rank nodes will become parents. When new nodes join the network, an optional stage, called *reverse direction*, is used. In this stage path information is collected as *Destination Advertisement Object (DAO)* messages travel from leaf (new nodes) to root (LBR). This information may be stored in LBR (non-storing mode) or parents (storing mode). Finally *maintenance* stage involves updating the DODAG (multicast tree) on a regular basis in the case that nodes move.

As an example, all pairs of nodes with lines between them in Fig. 1 are within wireless range (thin lines) of each other (i.e. have links). There is no link in the DODAG between 2 and 4 since node 4 chose node 3 as its parent (a node has only 1 parent). To deliver data in RPL, full flooding is used over the DODAG. Although there is no link in the DODAG from 2 to 4, when node 2 transmits, node 4 also receives this transmission due to the broadcast nature of wireless communications. Flooding in RPL causes many duplicate packets being received, which leads to higher loss of packets due to buffer overflow and collisions.

B. Reliabile Multicast with Trickle

Trickle Multicast (TM) adds control messages to reduce duplicate messages of RPL; it also provides reliable forwarding by using retransmissions. TM uses two mechanisms: data transfer with retransmission and consistency check to determine when to retransmit.

All nodes in the WSN continuously perform the Trickle Consistency Check (TCC). This involves each node sending a special ICMP packet to its 1-hop neighbors, who then respond.

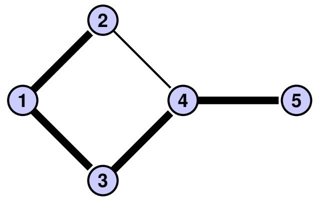

Fig. 1: **Bold lines:** DODAG graph with preferred parent. **Thin lines:** communication between node

The ICMP packets contain sequence numbers which allow a node to check the consistency of its current sequence number with its neighbors' sequence numbers.

When new data arrives at a node, that node increases its sequence number causing an inconsistency with its neighbors. An inconsistency means that the node must forward the data to the neighbors. If the data is received by the neighbors, they increase their sequence numbers thus returning to a consistent state. If however the data was lost (did not arrive at a neighbor), the node will learn of this via the regular TCC, i.e. sequence numbers are still inconsistent. This triggers the node to re-transmit the data to its neighbors, and assuming that it was received, returning to a consistent state.

The interval between TCC's is an important Trickle parameter. The interval is increased after a node has performed multiple successful checks. The interval is decreased if there are inconsistencies.

Trickle provides a mechanism to allow loss recovery, i.e. retransmissions when sequence numbers are inconsistent. One disadvantage of Trickle is that it may cause a large overhead, as original data packets, retransmitted data packets and TCC packets are sent. If the MAC protocol uses a radio duty cycle (e.g. ContikiMAC [8]), then there is an increased chance that packets will be lost. Another problem is that in a very high density network, radio collisions result in increased back-off at the MAC layer, which could especially increase the delay for TCC packets. Another problem with Trickle is that the delay from when an original data packet is sent and lost until when the retransmission occurs may be large. This is because a node only retransmits after a TCC. If the interval between TCC is large, the delay until retransmission is large.

C. SMRF

Stateless Multicast Forwarding with RPL in 6LowPAN Sensor Networks (SMRF) [6] was introduced to reduce the overhead of flooding in RPL. SMRF uses RPL storing mode, i.e. each node stores information about its DODAG parent. SMRF introduces two flooding optimizations: if a node receives a packet and that node has no DODAG children, then that node does not forward the packet to neighbors; and if a node receives a packet from a non-parent node, then again, that node does not forward the packet to neighbors. These selective flooding optimizations, illustrated in Fig. 3, aim to reduce the congestion in the network.

Another feature of SMRF is to take advantage of the radio-duty cycle used by WSN MAC protocols. To save power, a

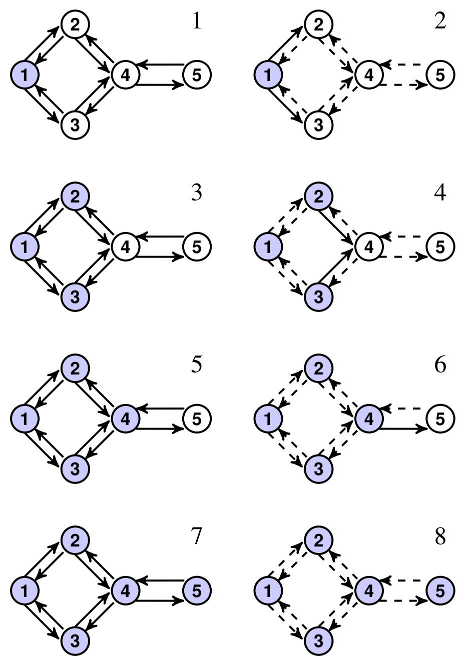

Fig. 2: Behavior of TM while disseminating a message. **Left:** TCC exchange **Right:** data dissemination after TCC **Dash arrow lines:** unwanted transmission

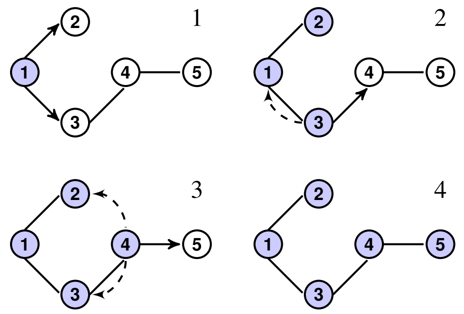

Fig. 3: Behavior of SMRF while disseminating a message. **Solid lines:** DODAG, **Dash arrow line:** unwanted transmission, **Arrow lines:** represent flow of forwarding by SMRF

node periodically switches its radio on and off. Packets can only be transmitted/received while the radio is on. A problem with this is that a destination node may have its radio off while the source node is transmitting a packet, causing packet loss. Therefore SMRF introduces the feature of informing the source node MAC to transmit a single packet multiple times. The MAC repeats the packet transmission while the radio is on, which increases the chance that the destination MAC will receive at least one copy of the packet.

[6] reports significant reduction in delay when using SMRF compared to TM, and higher packet delivery ratio in low density networks. However the transmission scheme of TM works better than SMRF (higher packet delivery ratio) in higher density networks.

III. iACK with RPL Forwarding

We have designed a new mechanism called *iACK* with the aims of increasing reliability compared to SMRF and reducing delay compared to Trickle while keeping the overhead minimal. iACK is based upon and uses similar mechanisms of SMRF. We introduce a retransmission mechanism in iACK. Section III-A explains the concept and the motivation of iACK's design. Then, the protocol operations are described in Sections III-B and III-C

A. iACK Concept

Typically, a wireless node in a IEEE 802.15.4 network transmits data through a broadcast medium (air) and a receiver will rebroadcast, and so on, until the data reaches every node in the network. Receiving a rebroadcasted packet (i.e. a packet that the node has already broadcast) acts as an implicit acknowledgement. Implicit acknowledgments appear in [9] and [10] as a reliability confirmation while a packet travels from sink to root node. They are also used in [11] and [12] in a Stop-And-Wait ARQ mechanism. Referring to Fig. 3 as an example, the dashed lines are implicit acknowledgments.

Although TM and SMRF use rebroadcasting, i.e. implicit acknowledgments, they do not exploit this information. Our contribution, which we call iACK, uses the implicit acknowledgments to improve the reliability of multicast communication in WSN. iACK consists of two components, flow propagation and retransmission protocol, described in the following sections. iACK relies on the following assumptions:

- Only one DODAG is used in the WSN at a time. Using two or more DODAGs at the same time will likely increase the congestion in the network, making the benefits of iACK minimal. Other approaches may be needed if multiple DODAGs are required.
- Nodes have enough memory to store previous packets. (In this paper, we restrict the number of previous packets to 8. Future work is needed to consider optimal values).
- Nodes are either static or have low mobility such that the DODAG does not change too often. E.g. a DODAG is fixed for duration that is much longer than the neighbor propagation period presented in Section III-B.

B. iACK Retransmission mechanism

The retransmission scheme in iACK consists of four processes described as follows.

Neighbor propagation. A node may receive rebroadcasts from multiple neighbors. In iACK, only neighbors which are its DODAG children will treat these rebroadcasts as implicit acknowledgments. Normally a node does not store the addresses of its DODAG children. Therefore a node needs to learn which nodes are its children. The procedure that iACK uses is: for each packet received, record the address of the sending node if the sending node:

1) is in the multicast group (DODAG), and
2) is not the preferred parent, and
3) has already sent at least min_neigh_msgs messages.

Each node therefore builds a list of children which is used in the next processes. This list can be updated, including entries deleted if communication is lost for some duration. The reason for updating the list of children is to recover if a node is moving away and/or a new node joins the DODAG; updates will allow iACK to work with the up-to-date DODAG.

Retransmission list management. A node that has built a list of children will maintain a buffer of packets to retransmit. The information about these packets is stored in a separate *retransmission list*. Fig. 4 shows how iACK manages the buffer and retransmission list. Packets received from the parent, which are to be forwarded to children, are added to the buffer. When the buffer is full (denoted by max_pkt_slot packets), a new packet overwrites the oldest packet.

For each packet in the buffer, and for each child that packet has been sent to, the retranamission list stores the count of retransmissions of that packet. When an implicit acknowledgement is received, the packet that it acknowledges is deleted from the retransmission list. In addition, if a packet is removed from the buffer or has been retransmitted the maximum allowed times (max_reTx), then the entry for that packet is deleted from the retransmission list.

Retrasmission. iACK periodically checks, every $reTx\_int$ seconds, whether a retransmission is needed (i.e. there is an entry in the retransmission list). The oldest packet in the retransmission list (i.e. the packet with the lowest sequence number) is scheduled for retransmission. There is a small delay of $reTx\_delay$ seconds before the retransmit. The reason for scheduling a retransmit in the future (as opposed to immediately retransmitting) is to allow the scheduler to find a time to transmit when the node is not transmitting/receiving other packets.

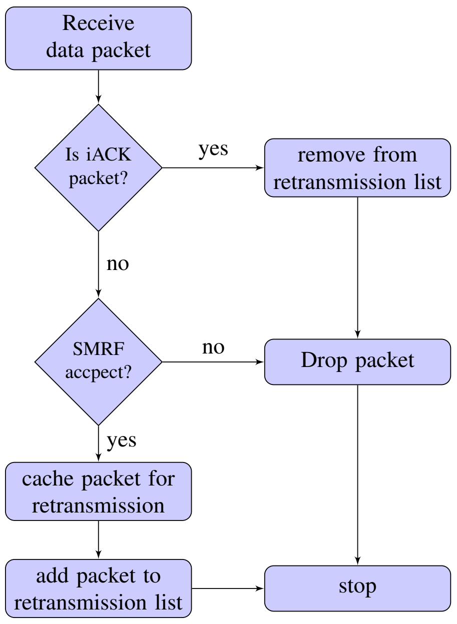
Receive data packet
Is iACK packet?
yes
remove from retransmission list
no
SMRF accpect?
no
Drop packet
yes
cache packet for retransmission
add packet to retransmission list
stop

Fig. 4: Process while node receive data packet

C. iACK Flow Propagation

As data is generated at the LBR it is sent along the first hop of the DODAG. As iACK (and both SMRF and TM) limit the sending rate of new data packets, if the data generated at the LBR is high, then many packets may be dropped across the first hop. To alleviate this effect, we used both explicit and implicit acknowledgements across this first hop. The LBR must cache extra packets for retransmission, increasing its memory requirements. However this should be acceptable in most cases as the LBR is often a powered device.

IV. Performance Evaluation

We implemented iACK in Contiki OS [13]. The modified code is based on the multicast branch provided by Geroge Oikonomou [14]. Then we simulated iACK, as well as Trickle and SMRF, in Cooja simulator [15] with instant-contiki 2.6. The parameters used in our simulations are listed in Table I. A straight line (chain) topology is used to investigate the impact of network size (hop length) on the algorithms. Traffic is generated at a constant rate (1 packet per second) but with varying loss rates as we are interested comparing the recovery features of the algorithms. in Future work will consider more realistic topologies and other parameter values. Unless otherwise stated, parameters for Trickle and SMRF are the default values used in [14]. Each simulation was repeated with 15 random seeds, and averaged results for the following metrics are recorded: data receive ratio, packet delay (both end-to-end and hop-to-hop) and packet receive count. Results and discussion are presented in the following sections.

TABLE I: Simulation set up parameter

Nodes	20 TMote Sky (1 source, 19 sinks)
Topology	Straight line
MAC Layer	IEEE 802.15.4
Duty Cycling	ContikiMAC
Seed	15 per each parameter permutation
Duration	1000 packets (20-30 min. real time)
Traffic Pattern	1 packet/sec, CBR
Data Flow	Unidirectional
Message Size	4 bytes at application layer
Transmit range	50 m
Interfere range	50 m
Distance Each Hop	46±4 m
min_neigh_msgs	5 times
max_reTx	4 times
max_pkt_slot	8 packets
reTx_int	0.5, 1 sec.
reTx_delay	uniform random(32,64) ms
Loss rate	0-50% (5% per increments)

A. Data Receive Ratio

Fig. 5 shows the data receive ratio at different nodes in the network when the loss rate is 0. The data receive ratio at a particular node is calculated as the percentage of packets received by that node relative to all 1000 packets transmitted by the source. Fig. 6 shows the data receive ratio averaged across all nodes for different loss rates.

We observe from Fig. 5 that TM has a rapid drop in data receive ratio after node 2. That is, packets are delivered

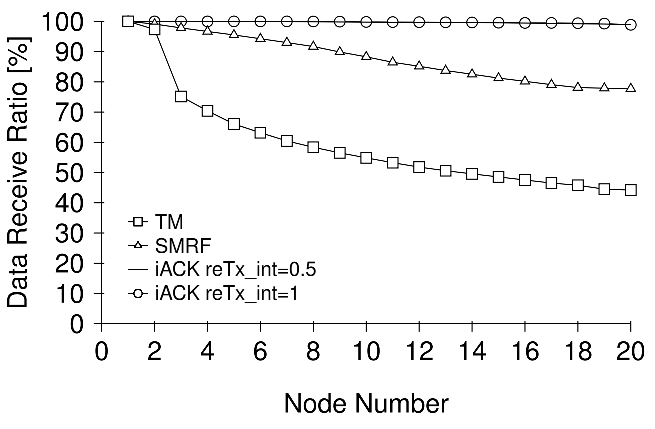

Fig. 5: Data Receive Ratio while hop increase

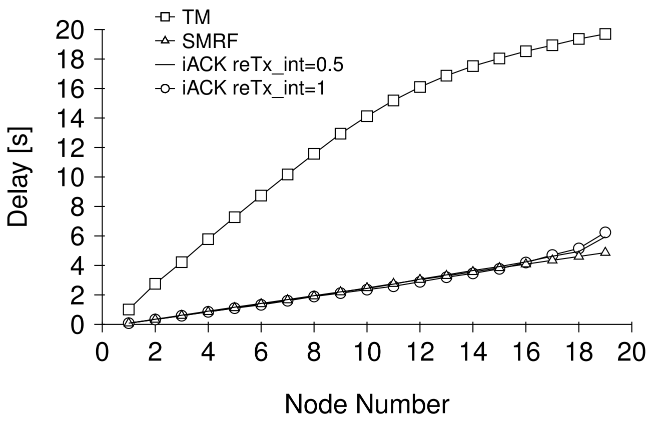

Fig. 7: Delay trend while hop increase

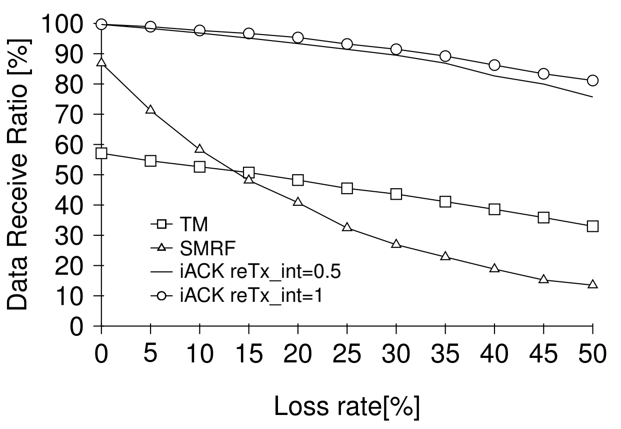

Fig. 6: Data Receive Ratio while loss rate increase

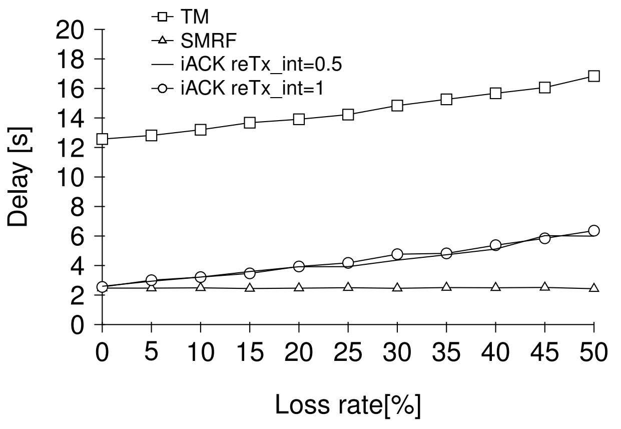

Fig. 8: Delay trend while loss rate increase

from root to node 1 successfully, but there are many dropped packets across subsequent hops. This is because the arrival rate of packets into the networks is greater than the rate that TM can transmits [6]. This is the motivation of the iACK flow propagation mechanism. Note that due to the TCC when the loss rate increases, the average data receive ratio only decreases slightly. The robustness of TM compared to SMRF, which has a rapid drop in data receive ratio in Fig. 6, is clear.

iACKs flow propagation and retransmission scheme lead to very high data receive ratio. Even with a loss rate of 50%, iACK achieves around 80% data receive ratio (while TM is around 40% and SMRF at 20%). The data receive ratio, as illustrated in Fig. 5 and Fig. 6, is a key strength of iACK compared to TM and SMRF.

B. End-to-End delay

Fig. 7 shows the delay from source to different nodes in the network when the loss rate is 0. Fig. 8 shows the average delay across all nodes for different loss rates.

We observe from Fig. 7 a rapid increase in delay when using TM. This is due to nodes having to wait for the TCC. With loss rate greater than 0 (Fig. 8), this is even worse as some TCC messages may be lost, causing nodes to wait longer.

SMRF has the smallest delay of the three schemes. There is only a small delay from when SMRF receives a packet until it is rebroadcast. And when packets are lost, although data receive ratio drops, there is no retransmission delay in SMRF.

iACK adds a retransmission scheme to SMRF. With no packet loss, it is almost equivalent to SMRF. However with packet loss, an increasing delay is observed, due to:

1) interval between checking ($reTx\_int$),
2) scheduled delay for each retransmission ($reTx\_delay$),
3) rate at which implicit acknowledgements are received.

Feedback receive rate in a network becomes a problem when a sender cannot receive iACKs. This may be due to iACKs being lost or because nodes do not generate iACKs (e.g. the leaf node in the DODAG). The lack of iACKs triggers the sender to not delete that particular packet from retransmission list, causing the sender to retransmit useless packets. The results show however that decreasing *reTx_int* can reduce iACK delay. Further analysis is needed to determine the optimal value of *reTx_int* and the impact of *reTx_delay*.

C. Packet receive counts

Fig. 9 shows the Packet Receive Count for different loss rates. This is the number of packets received by all nodes during the simulation. We group packets into three types:

- **control:** topology maintenance, e.g. DIO, DAO

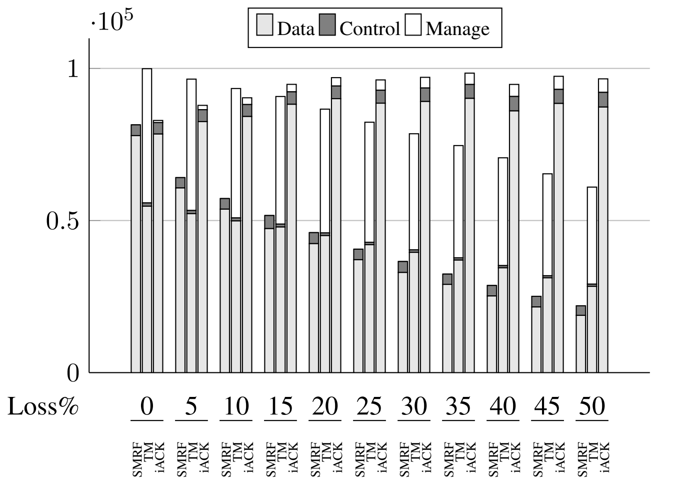

Fig. 9: Packet Receive Count categorize by forwarding method

- **manage:** protocol operation, e.g. custom ICMP, TCC
- **data:** a data packets

We observe that iACK has a high packet receive count, which indicates congestion. But note that the majority of the packets received are data, which gives a high data receive ratio. Compare this with TM, which also has a high packet receive count, however about half of the packets are management packets, i.e. TCC packets. TM wastes time transmitting TCC, while iACK is more efficient in sending data.

Both TM and iACK may cause congestion in the network. In the case that multiple DODAGs are to be supported, this can have a significant negative impact on data delivery ratio. SMRF may be better if multiple DODAGs are needed.

D. Memory

A drawback of iACK is that it requires buffer space to store multiple packets. This increases memory usage on each node when compared to both TM and SMRF.

The amount of buffer memory used is max_pkt_slot multiplied by the size of a packet (upto 127 bytes per packet). In many cases we believe this will be acceptable. However it may cause a problem when a network has multiple DODAGs or a high density if nodes. [6] has analyzed the memory usage of SMRF and TM. Further analysis is needed to compare iACK memory usage with SMRF and TM, especially considering different values of max_pkt_slot.

V. Conclusion

Reliable multicast is important for WSNs. Two current approaches are Trickle Multicast, which uses a retransmission scheme that leads to high delivery ratio but high delay and congestion, and SMRF, which is fast but does not offer high reliability. We have contributed a new mechanism for retransmissions, called iACK, that is a middle-point between TM and SMRF in performance. iACK takes advantage of rebroadcasts by children nodes, treating them as implicit acknowledgements. We designed a retransmission scheme that uses the implicit acknowledgements to determine which packets to retransmit. Simulation results show iACK has considerably higher data delivery ratio compared to both SMRF and TM, and lower delay than TM (and only slightly larger than SMRF). A key design point of iACK is that parameters can be adjusted to select an appropriate tradeoff between delay and delivery ratio depending on the scenario. A drawback of iACK is that it requires more memory in each node. Further analysis is needed to evaluate the memory usage and the performance of iACK in different topologies.

Acknowledgment

This research is financially supported by Thailand Advanced Institute of Science and Technology, National Science and Technology Development Agency, Tokyo Institute of Technology and Sirindhorn International Institute of Techcnology, Thammasat University and the National Research University Project, Thailand Office of Higher Education Commission. We thank George Oikonomou for providing Contiki code for Trickle and SMRF. We also thank Yamaoka Katsunori for continuous feedback on this research.

References

[1] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks." RFC 6550 (Proposed Standard), Mar. 2012.

[2] J. Tripathi and J. De Oliveira, "Proactive versus reactive revisited: Ipv6 routing for low power lossy networks," in *Information Sciences and Systems (CISS), 2013 47th Annual Conference on*, pp. 1–6, 2013.

[3] "Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans)," *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pp. 1–314, 2011.

[4] M. A. Mahmood and W. Seah, "Reliability in wireless sensor networks: Survey and challenges ahead," in *Preprint submitted to Elsevier*, pp. 1–41, 02 2012.

[5] R. K. J. Hui, "Multicast protocol for low power and lossy networks (mpl)," 08 2013.

[6] G. Oikonomou, I. Phillips, and T. Tryfonas, "Ipv6 multicast forwarding in rpl-based wireless sensor networks," *Wireless Personal Communications*, vol. 73, no. 3, pp. 1089–1116, 2013.

[7] C. S. N. Kushalnagar, G. Montenegro, "Ipv6 over low-power wireless personal area networks (6lowpans): Overview, assumptions, problem statement, and goals," 08 2007. RFC 4919.

[8] A. Dunkels, "The contikimac radio duty cycling protocol," Tech. Rep. T2011:13, Swedish Institute of Computer Science, Dec. 2011.

[9] M. Mahmood and W.-G. Seah, "Event reliability in wireless sensor networks," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2011 Seventh International Conference on*, pp. 377–382, 2011.

[10] K. Priya and S. Terence, "Retp: Reliable event transmission protocol in a wireless sensor network," in *Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN), 2013 International Conference on*, pp. 181–188, 2013.

[11] M. Maróti, "Directed flood-routing framework for wireless sensor networks," in *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '04, (New York, NY, USA), pp. 99–114, Springer-Verlag New York, Inc., 2004.

[12] L. Tuan, W. Hu, P. Corke, and S. Jha, "Ertp : Energy-efficient and reliable transport protocol for data streaming in wireless sensor networks," *Computer Communications*, vol. 32, pp. 1154–1171, May 2009.

[13] G. Oikonomou, "contiki-sensinode: mcast-forward." https://github.com/g-oikonomou/contiki-sensinode/tree/mcast-forward, 2012. [Lasted version: d10c64d Mar 13, 2012].

[14] A. Dunkels, "Contiki OS." http://www.contiki-os.org/, 2013. [Lasted version: Contiki 2.7 (15 November 2013)].

[15] F. Österlind and S. I. of Computer Science, *A Sensor Network Simulator for the Contiki OS*. SICS technical report, Swedish institute of computer science, 2006.