

Scalable Downward Routing for Wireless Sensor Networks and Internet of Things Actuation

Xiaoyang Zhong

Indiana University Purdue University
Indianapolis, Indiana, USA
Email: xiaozhon@iupui.edu

Yao Liang

Indiana University Purdue University
Indianapolis, Indiana, USA
Email: yaoliang@iupui.edu

Abstract—We present the **Opportunistic Source Routing (OSR)**, a scalable and reliable downward routing protocol for large-scale and heterogeneous wireless sensor networks (WSNs) and Internet of Things (IoT). We devise a novel adaptive Bloom filter mechanism to efficiently encode the downward source route in OSR, which significantly reduces the length of the source route field in the packet header. Moreover, each node in the network stores only the set of its direct children. Thus, OSR is scalable to very large-size WSN/IoT deployments. OSR introduces opportunistic routing into traditional source routing based on the parent set of a node's upward routing in data collection, significantly addressing the drastic link dynamics in low-power and lossy networks (LLNs). Our evaluation of OSR via both simulations and real-world testbed experiments demonstrates its merits in comparison with the state-of-the-art protocols.

Index Terms—Wireless sensor networks, Internet of things, downward routing, scalability, adaptive Bloom filter.

I. INTRODUCTION

In WSNs and IoT, data collection is the basic application scenario, where sensor nodes periodically sample and transmit data packets upward to one or multiple network sink(s). On the other hand, delivering control packets downward from the sink to individual sensor/actuator nodes is also essential for actuating target actuator(s), reconfiguring node parameters, or querying data from specific node(s). However, the WSN downward routing is significantly less studied than the upward routing. The major downstream protocols, such as Drip [15] and Glossy [4] are flooding based and disseminate control packets to the entire network, which is very inefficient in low-power, large-scale, and heterogeneous WSNs/IoT deployments. While the standard RPL, the IPv6 routing protocol for LLNs [17], offers the capacity of downward routing, it essentially suffers from the severe scalability problem for downward routing [1], [6]. Although approaches such as ORPL [3] and CBFR [12] attempted to address the scalability issue, their improvements are limited regarding highly resource-constrained wireless devices (see [7], for example).

In this work, we present the **Opportunistic Source Routing (OSR)** protocol to achieve desirable scalability and reliability for heterogeneous WSNs/IoT actuation, where upward routing topology tomography [9] is used for downward routing paths. OSR introduces opportunistic routing into the source routing based on the parent set [10] of a node's upward routing, to

exploit alternative downward paths to address wireless link dynamics. We devise a novel adaptive Bloom filter mechanism to efficiently encode and compress the source route path. The probabilistic nature of the Bloom filter passively enables opportunistic routing for downward packet forwarding. In addition, when a downward link between a parent node and its child node is broken, active opportunistic routing is activated to find one or more other parent(s) in the child's parent set for the downward forwarding. OSR only requires each node to store its *direct child set* rather than its *entire subgraph of descendants* as in RPL (storing mode) or in ORPL (compressed entire subgraph); therefore, OSR is extremely scalable for resource-constrained WSNs/IoT.

II. OSR DESIGN

The core mechanisms of OSR include path representation, direct child set, and opportunistic routing.

A. Adaptive Bloom Filter for Path Encoding

To make source routing scalable in WSNs/IoT, path encoding is a necessity given that a maximum IEEE 802.15.4 link layer frame is only 127 bytes. OSR exploits the Bloom filter [13] to encode a source route path in the packet header instead of storing the raw path.

Bloom filter [13] is a space efficient probabilistic data structure that supports insertion and membership query. To insert an element into a Bloom filter of m bits, k independent hash functions are applied to deterministically generate k hash values $h_i \in \{0, 1, \dots, m-1\}$, and the corresponding bits are set to 1 in the Bloom filter. For membership query, the queried element is hashed using the same set of hash functions. If all the k bits are matched in the Bloom filter, the element is considered being included/matched. A membership query may result in false positive, but never in false negative. The false positive (FP) rate of a Bloom filter can be calculated as following [13]:

$$p = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k, \quad (1)$$

where m is the length of the Bloom filter in bits, k is the number of hash functions, and n is the number of elements encoded in the Bloom filter.

Since the path length varies for each downward packet, we devise an adaptive path Bloom filter whose length m (in bits) is proportional to the hop count H of the route:

$$m = \begin{cases} 8H & H \leq L \\ 8L & H > L \end{cases}, \quad (2)$$

where L is the maximum Bloom filter length (in bytes) of any encoded source route. With a minimum node address length of two bytes, the devised Bloom filter (2) for path encoding leads to at least 50% space saving compared to the use of raw source route in RPL (non-storing mode).

For each downward packet, its source route is encoded in a Bloom filter by ORing the Bloom filters of all the node addresses in the source route. Upon reception of a downstream packet, a node performs membership query for each of its direct child, which can be done efficiently by an AND operation. If the check of any direct child is positive, the packet is forwarded downward to the matched child node(s).

B. Direct Child Set

Existing approaches such as RPL [17] (storing mode), CBFR [12], and ORPL [3] require each node to store/encode its entire subgraph of descendants for downward routing, causing the inherent scalability problem for large WSNs/IoT. In contrast, OSR only requires each node to store its one-hop direct children, referred to as the direct child set, for downward routing. Therefore, OSR is scalable with respect to node's memory.

OSR takes advantage of the underlying data collection protocol to establish the direct child set. When a node receives an upstream packet, it inspects the packet header and adds the sender's address to its direct child set. To address the link dynamics, each direct child is associated with a time-to-live (*TTL*) flag, which decreases periodically and resets upon packet inspection. When a child's *TTL* reaches 0, the child will be removed from the parent's direct child set.

C. Opportunistic Routing

During data collection, a node may have multiple candidate parents to forward its data packets, forming a parent set [10] of the node. The nodes in the same parent set have a high probability being in the transmission range of each other. Based on these observations, OSR introduces opportunistic routing (passively and actively) into the traditional source routing to improve the reliability of downward routing in dynamic WSNs/IoT.

Fig. 1 illustrates how opportunistic routing is conducted in OSR. The passive opportunistic routing, a consequence of the false positives of Bloom filter, is shown in Fig. 1a. The source route specifies $[\dots P \rightarrow C_1 \rightarrow T]$. Assume that nodes C_1 , C_2 , and C_3 are children of node P which are matched in the path Bloom filter. In addition to $(C_1 \rightarrow T)$, node C_2 is also in the parent set of node T , hence $(C_2 \rightarrow T)$ is an alternative path which can be exploited through the passive opportunistic routing. The active opportunistic routing is illustrated in Fig. 1b. Node T is the child of node P in

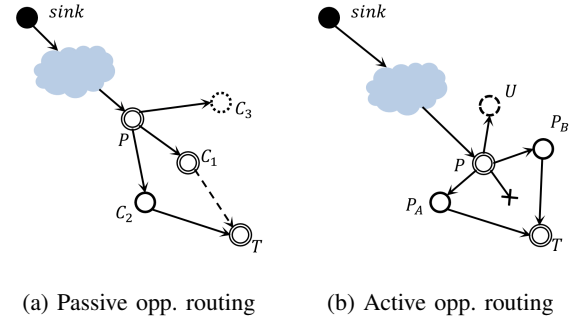


Fig. 1: Illustration of (a) passive opportunistic routing and (b) active opportunistic routing in OSR. The source route is marked as circles with double line border.

the source route. When P fails to deliver the packet to T , it broadcasts the packet to its neighbors. Three neighbors have received the broadcast; whereas neighbor U is not in the parent set of T and will ignore the packet, neighbors P_A and P_B will forward the packet to T because they are in the parent set of T . Thus, the obsolete link from P to T can be successfully bypassed by the active opportunistic routing issued by node P .

D. Downward Routing Decision

Unicast is the basic MAC transmission scheme used in OSR to deliver a downward packet. If any unicast fails after its maximum retransmissions, broadcast is used for active opportunistic routing. If a node has multiple direct children included in the source route, it uses local multicast to deliver the packet to all the matched children. Multicast is realized using broadcast if the MAC lacks the support (i.e., in TinyOS), hence requiring the receiving node to check its membership.

We have devised Algorithm 1 for downward forwarding at each intermediate node. Each node keeps a history of recently received downstream packets to detect and drop duplicates. A time-to-live (*TTL*) field is associated with each packet to avoid infinite forwarding loops.

III. EVALUATION

We implemented OSR in TinyOS 2.1.2, working in concert with CTP [5]. We performed a series of simulations and real-world WSN testbed experiments to evaluate OSR against ContikiRPL (both storing and non-storing modes) [16], TinyRPL (storing mode only) [14], and ORPL.

A. Evaluation in Cooja

We first conducted simulations in Cooja [11] using TelosB platform to evaluate the scalability of the protocols.

Inspired by [7], we evaluated the scalability of the protocols in a quasi-linear network topology with small twigs, which may be quite common for smart cities. The adopted quasi-linear network consisted of 74 nodes and built up to 68 hops, with the sink being at one end (as illustrated in Fig. 2). We used the Unit Disk Graph Medium (UDGM) with exponential distance loss as radio model with a maximum link quality of 90%. A node sent upward data packets every 10 minutes

Algorithm 1 Downward OSR

```

1: path_bflt: the Bloom filter of the source route.
2: matched_count: number of matched children in
   path_bflt.
3: tx_type: the transmission type of the OSR packet.
4: if (packet is duplicate) then
5:   return
6: end if
7: if (tx_type is Multicast) then
8:   if (local ID is NOT included in path_bflt) then
9:     Ignore the packet and return
10:  end if
11: end if
12: Check children for match
13: if (matched_count > 1) then
14:   Multicast the packet
15: else if (matched_count > 0) then
16:   Unicast the packet
17:   if (unicast fails) then
18:     if (tx_type is not Broadcast) then
19:       Broadcast the packet /* Opportunistic routing */
20:     end if
21:   end if
22: else
23:   /* no matched children, ignore the packet */
24: end if

```

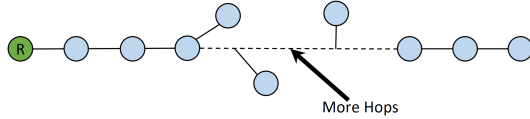


Fig. 2: The illustration of the quasi-linear topology.

on average. After 20 minutes network initialization, the sink started to send an actuation packet every 10 seconds to a randomly selected target node. The routing table size of ContikiRPL (storing) and TinyRPL was 50 since the TelosB RAM cannot contain all the 74 nodes. A total of 1320 downward packets were sent.

The results are shown in Table I. OSR has successfully reached all the nodes (up to the longest path of 68 hops) along the quasi-linear topology with 99.86% downward packet delivery ratio (PDR). In contrast, all RPL implementations suffer scalability problems. ContikiRPL (non-storing) only reached as far as 32 hops from the sink, far less than the RPL (non-storing) theoretical threshold of 64 hops and consequently had a poor PDR. On the other hand, the maximum reachable hop count in ContikiRPL (storing), TinyRPL, and ORPL was ad hoc, depending on the dynamics in the establishment of nodes' limited routing table. Their PDR performances were also significantly lower than that of OSR.

To better understand the protocols' scalability, we show in Fig. 3 the network PDR up to the first 25 hops in the down-

TABLE I: Scalability Comparison on Linear Network

Protocol	Max Reachable Hops	PDR(%)
OSR	68	99.86
ContikiRPL (NS)	32	33.31
ContikiRPL (S)	—	76.91
TinyRPL	—	27.95
ORPL	—	63.06

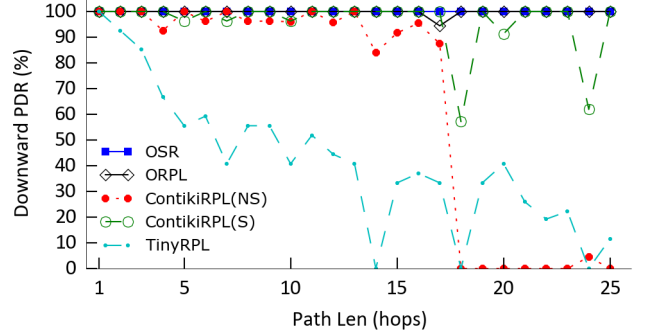


Fig. 3: Downward PDR for linear topology in the first 25 hops.

ward routing. The PDR of TinyRPL dropped quickly as the path length increased. ContikiRPL (non-storing) experienced a steep drop after the 17th hop, where the downward packet fragmentation began due to the long path length. ContikiRPL (storing) maintained high PDR for most of the time, but suffered from performance drop at several random hops, due likely to the probabilistic occupation in their ancestor nodes' routing tables at times, which was also observed for TinyRPL. ORPL achieved high PDR within the first 25 hops.

To summarize, ContikiRPL (non-storing and storing) and TinyRPL suffer from the scalability. IP fragmentation harms the performance of ContikiRPL (non-storing) significantly. ORPL also suffers from the scalability. In contrast, OSR scales significantly better than all the RPL implementations and ORPL.

B. Evaluation in Indriya

Next, we evaluate the reliability and energy efficiency of OSR in comparison with TinyRPL in the Indriya testbed [2]. ContikiRPL and ORPL were not included since they are based on the Contiki MAC layer which is very different from that of TinyOS platform. As we know, energy efficiency is heavily dependent on the platform in addition to the routing protocol.

The testbed was configured to low power with a sleep interval of 1 second using the default TinyOS MAC. Node 31 at the corner on the first floor was selected as the sink which was always on. The *MAX_BFLT_LEN* was 16 bytes.

Since TinyRPL could not work on the entire Indriya testbed, we conducted several experiment trials (30 minutes each) only using a half of the testbed (i.e., 47 nodes with odd IDs at the experiment time). The sink sent downward packets to a randomly selected node every 10 seconds after 10 minutes network initialization. A pure CTP experiment without any downward packets was conducted as the baseline.

TABLE II: Comparison between OSR and TinyRPL in Indriya

	PDR (%)	Duty Cycle (%)
CTP Baseline	—	4.26 ± 0.07
TinyRPL	89.45 ± 0.04	18.67 ± 0.45
OSR	97.80 ± 0.01	4.43 ± 0.17

TABLE III: OSR Performance in TOSSIM Simulation

WSN Size	PDR (%)	Max. Bftt (bytes)	Ucast Retx	Dup. Traffic ^a (%)	Pkt. with Opp. Routing	
					Active	Passive
225	98.67	16	49.15%	15.32	3.33%	13.33%
400	98.67	20	51.69%	17.92	4.33%	36.33%

^aWith regard to link layer transmissions.

Table II shows the comparison averaged on four trials. OSR (with CTP) performed significantly better than TinyRPL on both the downward PDR and the duty cycle (DC). TinyRPL's high duty cycle was mainly caused by its high DAO packet rate. We believe a careful tuning of the DAO rate could benefit TinyRPL's performance, however, it requires a systematic adjustment and is not the focus of this work. In particular, OSR itself only added a very little to the duty cycle compared with the CTP baseline, demonstrating its energy efficiency.

C. Evaluation in TOSSIM

We further conducted simulations using TOSSIM [8] to evaluate OSR for much larger network sizes and higher dynamics. Two uniformly distributed networks of sizes 225 and 400 nodes were generated with the sink at a corner. The network diameters were 11.58 hops and 19.07 hops, respectively.

The results are shown in Table III. Both tests achieved PDR above 98%, which was not affected by the expansion of the network. The link unicast retransmission ratio for both tests were around 50%, indicating a noisy and dynamic network condition. Regarding opportunistic routing, 3.33% and 4.33% of the packets experienced at least one active opportunistic routing occurrences (due to link failure) in the 225-node and 400-node simulations, respectively. On the other hand, we observed that for 225-node network, 13.33% of the downward packets have experienced passive opportunistic routing (due to false positives), whereas for the 400-node network the ratio was 36.33%, about 3 times of that in the 225-node network test, due to the longer downward path length. The opportunistic routing introduced about 15% to 18% duplicate traffic compared to the traditional source routing, which is inevitable due to the probabilistic nature of the Bloom filter (e.g., 9% to 50% duplicate traffic as in ORPL fig. 6(d) [3]).

IV. CONCLUSION

OSR provides reliable and scalable downward actuation in large-scale WSN/IoT systems. The unique opportunistic nature of OSR effectively addresses the drastic wireless link dynamics in noisy and resource-constrained LLNs. Effectively

encoding the source route and only storing the direct child set at each intermediate node, OSR has small memory overhead and achieves great scalability. The evaluation on both simulations and real-world WSN testbed experiments demonstrates that OSR significantly outperforms RPL both storing mode and non-storing mode on two most widely used implementations as well as ORPL. We believe OSR provides a significant and practical solution to wireless actuation for large-scale and resource-constrained WSN/IoT deployments.

ACKNOWLEDGMENT

This work is supported in part by NSF grant CNS-1320132.

REFERENCES

- [1] T. Clausen, U. Herberg, and M. Philipp, "A critical evaluation of the ipv6 routing protocol for low power and lossy networks (rpl)," in *2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2011, pp. 365–372.
- [2] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, "Indriya: A low-cost, 3d wireless sensor network testbed," in *Proc. of TridentCom*. Springer, 2011, pp. 302–316.
- [3] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree bloom: Scalable opportunistic routing with orpl," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '13. New York, NY, USA: ACM, 2013, pp. 2:1–2:14.
- [4] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *IPSN '11*. IEEE, 2011, pp. 73–84.
- [5] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "Ctp: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 10, no. 1, pp. 16:1–16:49, Dec. 2013.
- [6] O. Iova, P. Picco, T. Istomin, and C. Kiraly, "Rpl: The routing standard for the internet of things... or is it?" *IEEE Communications Magazine*, vol. 54, no. 12, pp. 16–22, December 2016.
- [7] T. Istomin, C. Kiraly, and G. P. Picco, "Is rpl ready for actuation? a comparative evaluation in a smart city scenario," in *European Conference on Wireless Sensor Networks*. Springer, 2015, pp. 291–299.
- [8] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 126–137.
- [9] R. Liu, Y. Liang, and X. Zhong, "Monitoring routing topology in dynamic wireless sensor network systems," in *ICNP '15*, Nov 2015, pp. 406–416.
- [10] M. Navarro and Y. Liang, "Efficient and balanced routing in energy-constrained wireless sensor networks for data collection," in *EWSN '16*, USA, 2016, pp. 101–113.
- [11] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, Nov 2006, pp. 641–648.
- [12] A. Reinhardt, O. Morar, S. Santini, S. Ziller, and R. Steinmetz, "Cbfr: Bloom filter routing with gradual forgetting for tree-structured wireless sensor networks with mobile nodes," in *WoWMoM '12*, June 2012, pp. 1–9.
- [13] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," *IEEE Communications Surveys Tutorials*, vol. 14, no. 1, pp. 131–155, First 2012.
- [14] TinyOS, *TinyRPL*, 2011. [Online]. Available: <http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyRPL>
- [15] G. Tolle and D. Culler, "Design of an application-cooperative management system for wireless sensor networks," in *Wireless Sensor Networks, 2005. Proceedings of the Second European Workshop on*. IEEE, 2005, pp. 121–132.
- [16] N. Tsiftes, J. Eriksson, and A. Dunkels, "Low-power wireless ipv6 routing with contikirlpl," in *IPSN '10*. New York, NY, USA: ACM, 2010, pp. 406–407.
- [17] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander, "Rpl: Ipv6 routing protocol for low-power and lossy networks," *RFC6550*, 2012.