Recycling Corrupt Packets over Multiple Hops

Muhammad Hamad Alizai⋆, Muhammad Moosa Khattak⋆, Dong Han†, Omprakash Gnawali†, and Affan A. Syed‡

⋆University of Engineering and Technology, Peshawar, Pakistan
†Department of Computer Science, University of Houston, USA
‡National University of Computer and Emerging Sciences, Islamabad, Pakistan
{hamad.alizai,moosa.ktk}@uetpeshawar.edu.pk, {donny,gnawali}@cs.uh.edu, affan.syed@nu.edu.pk

Abstract. We propose a Corrupt Packet Recycling (CPR) approach for WSN that processes and forwards partially-corrupt packets over multiple hops without necessitating their complete recovery. We motivate this approach with two insights: address-agnostic routing in WSN can forgive header errors since intermediate nodes know the next hop and the destination; and that payload errors can be either interpolated, due to error-tolerant nature of information in WSN applications, or rectified using spatio-temporal redundancies. CPR, without introducing any transmission overhead, improves information delivery rate by up to 4×.

Keywords: error tolerant, multihop wireless, data collection

1 Introduction

Partially-corrupt packets contain valuable information. Since most corrupt packets have only few symbol-errors [3,11], discarding such packets results in wasteful information loss and reduced network efficiency in terms of reliability, latency, energy and bandwidth. Techniques such as partial-packet recovery [4] and error checksums [5] try to recover corrupt packets at a smaller cost than traditional ARQ but have three limitations: First, they can only repair a small subset of corrupt packets locally (i.e., with multiple copies) [1] discarding others for which bit-by-bit information is not recovered. Second, they introduce undesirable transmission overhead in WSN, e.g., 22-64% for FEC [5] and preamble-header duplication for partial recovery [4]. Third, they operate at PHY and link layers and do not consider the potential for packet recovery over multiple hops.

We propose an error-tolerant approach at the network layer, called CPR (corrupt packet recycling), that *recycles* partially-corrupt packets over multiple hops towards the base station. The term recycling refers to *processing and forwarding corrupt packets without necessitating their complete recovery* at intermediate nodes. CPR achieves this recycling by providing a simple, best-effort service to locally repair header errors while concealing payload errors altogether during multihop communication. As a result, CPR is the first approach (i) that can recycle 100% corrupt data packets[1] received at an intermediate node, (ii) has no transmission overhead, and (iii) and operates at network layer to facilitate multihop operation.

CPR is enabled by two key insights of WSN characteristics: First, *adress-agnostic hierarchical routing* (e.g., collection tree) results in each data packet forwarded to a single or same set of outgoing links. This allows CPR to easily tolerate "header errors" since, once the communication paths are established, each intermediate node antecedently knows the next hop and the destination. Second, we see that the *nature of information in WSN is often error-tolerant*: data bytes are typically a digital quantization of analog signals from potentially inaccurate and heterogeneous sensors. Thus bit-level "payload errors" are often tolerable by WSN applications as compared to TCP/IP based networks where fidelity of content (like files and video) remains essential. For example, in some applications [7, 8, 12], it is useful to receive a packet even if it is corrupt because reception itself could provide information about the sensing activity. Furthermore, base station can utilize state-of-the-art data mining techniques or spatio-temporal redundancies in transmissions (due to dense and overlapping node deployments) to extract meaningful information from erroneous data.

The main goal of CPR is thus to avoid data packet drop "no-matter-what". For this purpose, CPR enables wireless protocols at the network layer to handle corrupt packets in two steps. The first step is *header recovery* using domain knowledge obtained from a history of correct packets from the same source. For example, in a typical collection tree, if the `origin` field in the packet header is undamaged, the packet `sequence-number` can be recovered based on the transmission frequency of previous packets from the same source. The second step is *forwarding* of packet, whose header has been recovered in the first step, to the next hop in the collection tree. However, CPR can also be "stubborn" in the second step, i.e., forward packets even if the header recovery failed to achieve recycling of 100% network-level packets.

Contributions: (i) We motivate and design an error-tolerant approach at the network layer that recycles corrupt packets over multiple hops (Section 2 and 3). (ii) Our preliminary results — up to 4× improvement in information delivery rate — advocate the high utility of CPR for WSN (Section 4).

2 The Need for Recycling Packets in WSN

Figure 1 motivates the need for CPR: the number of partially-corrupt packets varies between $< 10\%$ to $> 70\%$ on just a single hop, across different deployment conditions. Thus, every recycled packet (i) saves retransmissions and improves latency, energy and bandwidth efficiency, and (ii) delivers valuable information that otherwise is completely lost due to packet drop. We see that a vast majority of WSN applications are indeed tolerant to corruption of individual bytes and thus can benefit from recycling packets over multiple hops. We now argue

[1] All network layer data packets that bypass link layer CRC.

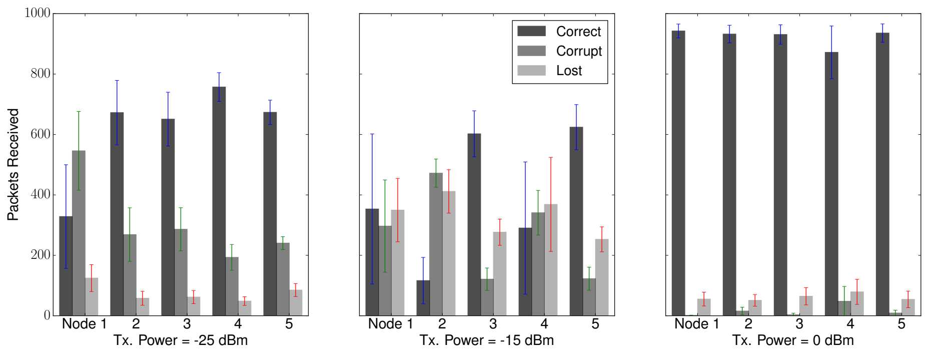


Fig. 1. Average packet reception over five experiments, in each we sent 1000 packets every 128ms, at five TelosB receiver nodes radially distributed (4.5m) around a sender indoors. The number of corrupt packets (mismatched CRC) vary over different deployment conditions depicted by tx power levels. Lost packets are not notified by the network interface. Error bars represent standard deviation.

this breadth of coverage by presenting how the two broad categories of WSN applications benefit from this approach.

Passive Collection Streams is the most common category of WSN applications with a goal to maximize the network lifetime of periodic data collection. However, most practical deployments of WSN, from the Great Duck Island [8] to Volcano deployment [12], report the need to revisit these deployments due to (a) insufficient amount of data collected at sink and (b) reduced network lifetime from excessive radio hardware utilization. CPR can help improve the information delivery rate for these applications while reducing radio activity, thus obviate the need for manual reprovisioning.

Another category of WSN —*Active Event Detection*— actively monitors the environment for an event of interest. These applications mostly remain in a quiescent state, generating none-to-very-little traffic, but have bursts of very critical data generated when an event is detected. We believe that, while the delivery of the actual data (a digitized sensor reading), is important, delivering partially-corrupt packets still conveys meaningful information to the application. Consider a fire-monitoring application that, on event detection, send a much higher rate of packets reporting the intensity of fire at a particular location. From the application perspective, even the increased rate of *possibly corrupt* packet-delivery is a good indication of an alarm condition that can trigger some appropriate response.

An orthogonal benefit in employing an error-tolerant approach becomes evident for WSN deployed in *extreme communication environments*, such as burrows, underwater, and industrial settings. Here, scientists are known to struggle in collecting information [7, 9] since most wireless links have poor quality. CPR can utilize such unreliable links which are otherwise rendered useless for packet forwarding. In a DTN based burrow deployment [7], it has been demonstrated that turning off CRC allows efficient neighborhood discovery without imposing correct packet reception.

These examples above motivate the case for recycling corrupt packet in WSN. We now focus on the design of one particular instance of CPR approach to facilitate its implementation in real-world and evaluate its benefits.

3 Designing an Instance of CPR

Using CPR, we can either develop a new routing system or upgrade an existing protocol. Here we focus on the latter to emphasize on the CPR approach itself rather than on protocol development related intricacies. When integrated with existing protocols, the design of CPR is strongly dependent upon the host protocol. Therefore, we first select a host protocol and then detail our customized solution.

3.1 The Host Protocol and its Header

We use CTP [2], a widely used collection protocol for experimentation, as the host protocol. We revisit each field in the CTP packet header to investigate if it is necessary for each field to be received correctly for successful packet delivery to the collection root. We find that an error in almost all these fields could be ignored or repaired to improve data delivery in the network. In our scheme, a *few* correctly received packets provide enough context to a node to repair these fields in partially-corrupt packets. Unlike single-hop error-tolerant techniques used in protocols such as Refector [10] and UDP Lite [6], our scheme can successfully fix and forward a corrupt packet over multiple wireless hops.

P: Routing pull (1 bit) - *The nodes use the P bit to request routing information from other nodes.* If this bit is flipped 0 to 1, we may have unnecessary control packets. If this bit is flipped 1 to 0, the control information may be delayed until correct reception in the future. In either case, we can continue to forward the data packet. We can use similar argument for the next four header fields — `congestion notification` (1 bit), `reserved` (6 bits), `THL` (1 byte), and `ETX` (1 byte) — as their integrity may be ignored temporarily to prevent dropping packets that can be successfully forwarded. `THL` and `ETX` are used to avoid loops which rarely occur in CTP [2].

Origin (2 bytes) - *The originating address of the packet.* This field is not essential for packet forwarding. An error in this field may cause a duplicate packet cache miss decreasing the effectiveness of duplicate suppression – a small price to pay instead of dropping a multihop packet. It is possible to recover this field in a corrupt packet by correlating its `length`, `ETX`, and/or `seqno` with a sample of correct packets from the same source.

Seqno (1 byte) - *Origin sequence number:* The `seqno` field is required to uniquely identify packets from the same source. This field can be recovered if the `origin` field in the packet is undamaged, for example, by observing the transmission frequency of previous packets from the same origin. Moreover, as for the `origin` field, sampled header information from previous successful packets can be used to recover this field.

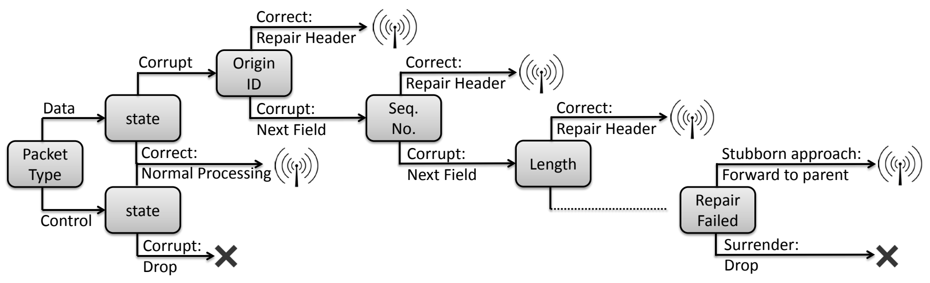


Fig. 2. The decision tree for repairing routing headers.

Collect_id (1 byte) - *Higher-level protocol identifier:* This field only needs to be recovered if CTP is serving multiple flows, which is not a frequent phenomena in application-centric WSN. Apart from the mechanism advocated for `origin` and `seqno` above, this field additionally provides the possibility of manually increasing Hamming distances (i.e., the number of differing bits between two bit strings) between `collect_id` of different flows. An incoming packet at the base station with a corrupt `collect_id` can be assigned to the flow with minimum hamming distance.

Data (max 128 bytes for IEEE 802.15.4) - *the payload.* This is the *don't care* part of the packet from CPR perspective since our goal is to conceal payload errors until packet reaches its intended destination.

Overall, we can conclude that errors in most header fields of a collection protocol can be repaired with the knowledge of the protocol and application and the tradeoffs we are willing to make. Important fields, such as `origin`, can be recovered if some of the header fields are correctly received.

3.2 Header Recovery Algorithm

Our algorithm is based on *decision tree* classification method which selects a class by descending a tree of possible decisions. Each internal node in the tree corresponds to one of the input variables. While descending the tree, at each node, the corresponding input variable is compared with a threshold value. This threshold value is determined based on training data, which, in our case, is a history of K correct packets from a particular origin. One of the two child nodes is then selected based on the result of the comparison until leaf node, that is the final prediction, is reached. For example, in a corrupt packet, we can determine whether or not the `origin` field (input variable) is corrupt by observing if the node has recently received packets from the same origin (threshold value from training data).

We use decision tree analysis because of its suitability for WSN: it is simple to implement, compute and respond when compared with alternative methods such as neural networks. One possible instance of CPR's decision tree is shown in Figure 2. The position of nodes in this tree can change based on the information gain from the training data, e.g., the `origin` node can be replaced by `length` node and vice versa.

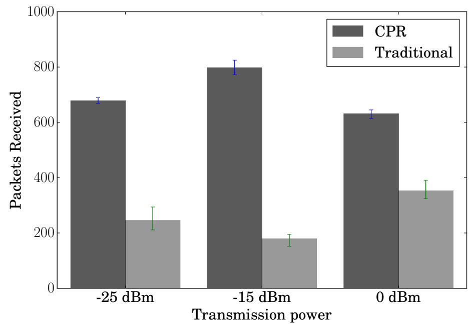


Fig. 3. Average (of three experiments) packets received at sink node, out of 1000 transmitted, by a sender node at one end of a linear five-hop topology with inter-node spacing of $4.5m$. CPR, with corrupt packets being recycled, outperforms the traditional approach. The error bars represent the best and worst of these experiments.

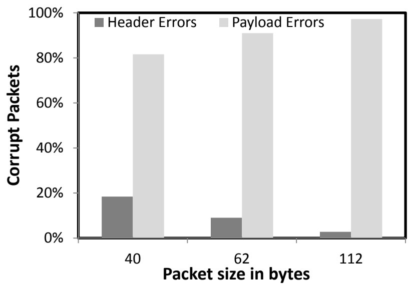
Please note that CPR only targets data packets. Corrupt control packets are immediately dropped as (i) they are typically not retransmitted and (ii) they carry vital information necessary to maintain a robust network topology. Apart from link layer de-multiplexing, corrupt control and data packets can be differentiated simply by using significantly different packet sizes.

4 Preliminary Results

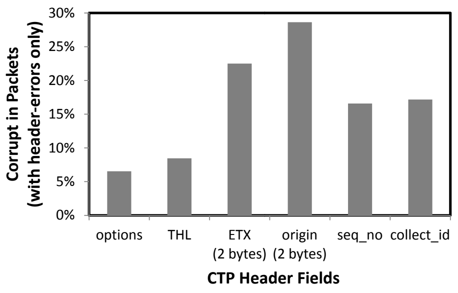
We now describe the results from experiments that evaluate the benefits of CPR in terms of header recovery and data delivery over traditional, non-recycling, multihop data collection (i.e., CTP [2]). Our comparison focuses on the traditional approach since CPR operates at the network layer and any link layer recovery techniques are orthogonal and will equally improve the performance of routing with and without CPR.

4.1 Multihop Packet Reception Rates

We first evaluate packet reception rates (PRR) over multiple hops to quantify the raw magnitude of improvement achievable with CPR. This implies (i) we use the "stubborn" forwarding mechanism of CPR, i.e., forward packets even if header recovery fails, and (ii) we disabled retransmissions in CTP so we can study the enhancement due to only CPR's mechanisms in a simple multihop protocol. Our experiment setup uses a linear topology of five hops where a sender, at one end of the topology, transmits one thousand packets to the root node at the other end over 5 hops. Figure 3 shows the results both for CPR, with intermediate nodes forwarding corrupt packets, and traditional, where the nodes forward only the packets with valid CRC. With CPR, we see 2-4$\times$ improvement in PRR, indicating a significant potential to improve throughput, latency and lifetime (by reducing the need for retransmissions).



(a) Header vs Payload errors using different packet sizes



(b) Header-field error distribution only in packets with corrupt CTP headers.

Fig. 4. Recycling efficiency of CPR.

It is clear that improvement in PRR due to recycling depends upon the number of corrupt packets in a particular deployment. Hence, to maximize the magnitude of this improvement, the main goal of CPR is to ensure maximum recycling at an intermediate node irrespective of deployment conditions. We next evaluate this recycling efficiency of CPR.

4.2 Network Layer Recycling and Header Recovery

We estimate the number of corrupt packets that are recycled at an intermediate node (i) with successful header recovery, and (ii) with header errors (head recovery failed). For this purpose, we first need to determine the number of packets that are received with only header errors. With symbol-errors frequently reported as roughly uniformly distributed in corrupt packets [3,11], we can easily estimate the ratio of header vs payload errors for a particular packet size. However, here we calculate this ratio through empirical observations.

We place two motes 4.5m apart. For each packet size (see Figure 4(a)), the sender transmits packets with an interval of $128ms$. We stop the experiment when the receiver receives 1000 corrupt packets for a particular packet size. Our results in Figure 4(a) substantiate the roughly uniform distribution of symbol errors in packets: for 40 bytes packets (with 8 bytes header for CTP), nearly 20% of the packets have corrupt headers. Similarly, the header errors are less likely as payload size increases. We can thus conclude that for the worst case of the presented data (size = 40 bytes), 80% of the packets will be automatically recycled in CPR with correct headers.

We now narrow down our focus on the remaining 20% of packets with only corrupt headers. Figure 4(b) shows how the corruption is distributed across different header fields for these packets. To understand the worst case performance of CPR, we assume that only the headers with the correct `origin` field can be recovered[2]. As shown in the Figure 4(b), 30% of the these packets have corruption in the `origin` field suggesting 70% headers can be recovered in the worst case.

[2] Our estimate is deliberatively conservative because we want to do worst case analysis.

Overall, in this scenario, CPR recycles two sets of packets with correct headers: (i) 80% of the total packets with only payload errors (see Figure 4(a)), and (ii) 14% of the total packets (0.7 × 20% with correct `origin` in Figure 4(b)) whose headers are repaired by CPR algorithm. Hence, in the worst case, CPR recycles 94% of the corrupt packets with no header errors. The remaining 6% can be recycled using "stubborn" forwarding.

5 Conclusions and Future Work

This paper presents a network layer error-tolerant approach for recycling corrupt packets over multiple hops. CPR, by avoiding packet drop, improves information delivery that can benefit error-tolerant WSN applications. Our preliminary evaluation demonstrate the high utility of CPR in terms of information delivery (2-4× improvement) and recycling efficiency per intermediate node (100%).

A complete implementation of header recovery algorithm and thorough evaluation on a widely used testbed is still pending. An important implementation aspect is to enable the link layer to issue acks for corrupt packets to avoid retransmissions. We also plan to extend and explore CPR's utility for 6LoWPAN and multi-radio systems.

References

1. H. Dubois-ferrière, D. Estrin, and M. Vetterli. Packet combining in sensor networks. In *Sensys*, 2005.
2. O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Sensys*, 2009.
3. F. Hermans, H. Wennerström, L. McNamara, C. Rohner, and P. Gunningberg. All is not lost: Understanding and exploiting packet corruption in outdoor sensor networks. In *EWSN*, 2014.
4. K. Jamieson and H. Balakrishnan. Ppr: Partial packet recovery for wireless networks. In *SIGCOMM*, 2007.
5. J. Jeong and C.-T. Ee. Forward error correction in sensor networks. In *WWSN*, 2007.
6. L.-A. Larzon, M. Degermark, S. Pink, L.-E. Jonsson, and G. Fairhurst. The Lightweight User Datagram Protocol (UDP-Lite). RFC 3828, 2004.
7. J. Á. B. Link, G. Fabritius, M. H. Alizai, and K. Wehrle. Burrowview - seeing the world through the eyes of rats. In *IEEE PerCom, Workshop Proceedings*, 2010.
8. A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM WSNA*, 2002.
9. M.Cattani, M.A.Zuniga, M.Woehrle, and K.G.Langendoen. Sofa: Communication in extreme wireless sensor networks. In *EWSN*, 2014.
10. F. Schmidt, M. H. Alizai, I. Aktaş, and K. Wehrle. Refector: Heuristic header error recovery for error-tolerant transmissions. In *CoNEXT*, 2011.
11. F. Schmidt, M. Ceriotti, and K. Wehrle. Bit error distribution and mutation patterns of corrupted packets in low-power wireless networks. In *WiNTECH*, 2013.
12. G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 2006.