ARQ for Physical-layer Network Coding
Jianghao He and Soung-Chang Liew, Fellow, IEEE

Abstract-This paper investigates ARQ (Automatic Repeat request) designs for PNC (Physical-layer Network Coding) systems. We have previously found that, besides TWRC (Two-Way Relay Channel) operated on the principle of PNC, there are many other PNC building blocks—building blocks are simple small network structures that can be used to construct a large network. In some of these PNC building blocks, receivers can obtain side information through overhearing. Although such overheard information is not the target information that the receivers desire, the receivers can exploit the overheard information together with a network-coded packet received to obtain a desired native packet. This leads to throughput gain. Our previous study, however, assumed what is sent always get received. In practice, that is not the case. Error control is needed to ensure reliable communication. This paper focuses on the use of ARQ to ensure reliable PNC communication. The availability of overheard information and its potential exploitation make the ARQ design of a network-coded system different from that of a non-network-coded system. In this paper, we lay out the fundamental considerations for such ARQ design: 1) We address how to track the stored coded packets and overheard packets to increase the chance of packet extraction, and derive the throughput gain achieved by tracking 2) We investigate two variations of PNC ARQ, coupled and non-coupled ARQs, and prove that non-coupled ARQ is more efficient; 3) We show how to optimize parameters in PNC ARQ—specifically the window size and ACK frequency—to minimize the throughput degradation caused by ACK feedback overhead and wasteful retransmissions due to lost ACK.

Index Terms—Physical-layer Network Coding, Multi-hop Wireless Networks, Selective Repeat ARQ, Selective Acknowledgements, Wireless Scheduling, Overheard information, End-to-End Design, Unicast.

I. INTRODUCTION

Physical-layer network coding (PNC) [1] has been shown to outperform the traditional wireless relaying schemes. In a two-way relay channel (TWRC), by allowing the two end nodes to transmit simultaneously to the relay and not treating this as collision, PNC can boost the system throughput by 100% [1]. Furthermore, as shown in [2], besides TWRC, PNC can also be applied to many other relay network structures, referred to as PNC building blocks (or PNC atoms). In essence, a large network can be decomposed into a multiplicity of PNC building blocks so that PNC can be applied in a systematic way to achieve throughput gain. Each of these building blocks can achieve throughput gain ranging from 50% to 167%, translating to overall throughput gain of around 100% for the large network [2].

The throughput gain as demonstrated in [2] assumes transmissions in the building blocks are reliable, that is, all transmissions are successfully received. In practice, since these transmissions are over error-prone wireless links, the packets may get corrupted and not received. Automatic repeat request (ARQ) is a means for error control to ensure reliable communication. To our best knowledge, ARQ for PNC systems has not been systematically studied before. Specifically, ARQ that takes into account that PNC building blocks may exploit overheard packets to achieve throughput gain has not been studied. In this paper, we investigate the design principles of such ARQ in an attempt to answer the following questions:

1. What is the difference between PNC ARQ and traditional ARQ? What are the design al-
ternatives for PNC ARQ? How to improve the efficiency of the retransmissions in PNC building blocks that exploit overhearing?

2. What are the optimal throughputs of PNC building blocks with erasure channels? Can ARQ be designed to allow the PNC building blocks to achieve these optimal throughputs?

To illustrate ARQ for PNC building blocks with overhearing, let us give an example based on a specific PNC building block, referred to as a cross atom, as shown in Fig. 1. The cross atom consists of one relay, two source nodes, and two receiver nodes. There are two unicast flows, A-R-C and B-R-D. Node A wants to transmit a sequence of packets, \( A_1, A_2, A_3, \ldots \), to node C via relay R; node B wants to transmit a sequence of packets, \( B_1, B_2, B_3, \ldots \), to node D via R. Suppose that 1) nodes C and D can overhear the transmissions of nodes B and A, respectively; and 2) the transmissions of A and B do not affect the overhearing/reception of node B’s packets by node C and the overhearing/reception of node A’s packets by node D. If links are perfectly reliable, two time slots are needed to deliver one packet from A to C and one packet from B to D, as explained below. In the first timeslot, nodes A and B transmit packets \( A_1 \) and \( B_1 \), respectively. Nodes C and D overhear packets \( B_1 \) and \( A_1 \), respectively. With the PNC mechanism, node R receives packet \( A_1 \oplus B_1 \). In the second timeslot, node D fails to receive packet \( A_1 \oplus B_1 \), while node C successfully receives packet \( A_1 \oplus B_1 \) from relay R. After one round (the two timeslots), neither node C or D can extract its desired packet.

Now, suppose that each link is a packet erasure channel rather than a perfectly reliable channel. Suppose that in the first time slot, node C fails to overhear packet \( B_1 \), while node D successfully overhears packet \( A_1 \) and relay R successfully receives \( A_1 \oplus B_1 \). Furthermore, in the second time slot, node D fails to receive packet \( A_1 \oplus B_1 \), while node C successfully receives packet \( A_1 \oplus B_1 \) from relay R. After one round (the two timeslots), neither node C or D can extract its desired packet.
packet (packets $A_1$ and packet $B_i$, respectively). Thus, an ARQ mechanism is needed to ensure nodes C and D can eventually obtain packets $A_1$ and packet $B_i$.

In this paper, we focus on end-to-end ARQ. For the cross atom, this means that the ARQ mechanism is between the source nodes, A and B, and the destination nodes, C and D. The relay does not take part in packet retransmission and is oblivious of the ARQ mechanism: it just relays network-coded packets.

Continuing with the example, although nodes C and D could obtain only packet $A_i \oplus B_i$ and packet $A_1$, but not their desired packets, packet $A_i$ and packet $B_i$, respectively, the received packets are nonetheless useful in the future and should be stored. In this example, suppose we adopt an ARQ window size of one. Since nodes C and D did not receive their desired packets, no ACK will be returned to nodes A and B. In the next round, since nodes A and B did not receive ACKs, they retransmit packets $A_i$ and $B_i$, respectively, in the third timeslot (the first timeslot of the second round). Now, as long as node C can overhear packet $B_i$ in the third timeslot, it can already extract its desired packet $A_i$, since node C already has packet $A_i \oplus B_i$ from the last round. This is the case even if node C does not receive $A_i \oplus B_i$ from R in the fourth timeslot. By the same token, even if node D fails to overhear packet $A_i$ in the third timeslot, as long as it receives $A_i \oplus B_i$ from R in the fourth timeslot, it can extract packet $B_i$. The storage and the reuse of overheard and coded packets can increase the chance of destination nodes extracting their desired packets in a lossy network.

With the possibility of reusing stored overheard and coded packets, an issue that does not occur in traditional ARQ arises: how should we “track” which stored packets may still be potentially useful and which stored packets will no longer be useful in the future, especially if we increase the ARQ window size to beyond one? Specifically, we need a tracking mechanism to decide which overheard packets and coded packets should continue to be stored, and which should be discarded.

This stored-packet tracking raises yet another design issue, as explained below. In general, each PNC building block has multiple unicast traffic flows between several source-destination pairs. Consider one particular flow $f$. It is natural for the source node of $f$ to move on to deliver the next packet after its current packet has been successfully delivered. For example, in the cross PNC atom in Fig.1, after node A successfully delivers packet $A_i$ to node C, it will transmit packet $A_{i+1}$ in its next transmit. This strategy is referred to as non-coupled ARQ (its formal definition is given in Section III). It is also possible to have another strategy, referred to as coupled ARQ (formal definition also in Section III), wherein the source node of $f$ may not move on to deliver its next packet even after the current packet has been successfully delivered, because the current packet of a different flow $g$ has not been successfully delivered. This is the case, for example, if
flow $g$ wants to use the current packet of $f$ to extract a desired packet. With respect to the cross PNC atom in Fig.1, for example, if nodes A and B transmit packets $A_i$ and $B_i$, and node C successfully extracts packet $A_i$ but node D fails to extract packet $B_i$. Furthermore, suppose that the reason for the failure at node D is that node D only receives packet $A_i \oplus B_i$ but misses packet $A_i$. If node A progresses to transmit packet $A_{i+1}$, then the packet $A_i \oplus B_i$ will not be useful at node D. However, if node A retransmits $A_i$, then the packet $A_i \oplus B_i$ will still be useful. In coupled ARQ, the progress of the transmissions of different flows are coupled together, hence the term “coupled ARQ”. Compared with non-coupled ARQ, the tradeoff is that node A misses the chance to transmit a new packet. In this paper, we prove via a throughput analysis that non-coupled ARQ is more efficient than coupled ARQ.

In general, not only DATA packets, ACKs may also get lost in wireless networks. Missing ACKs leads to wasteful retransmissions (i.e., packets may be retransmitted even if their target receivers already have them). Transmitting excessive number of ACKs to ensure their receptions, on the other hand, may lead to large feedback overhead. This paper also investigates how to set parameters in the PNC ARQ mechanism—specifically the window size and the ACK frequency (i.e., number of DATA packets per ACK)—to strike a balance to optimize throughput.

In summary, to answer the first set of questions as set out at the beginning of this section:

1) We address the importance of tracking the stored coded packets and the overheard packets, a distinguishing feature that sets PNC ARQ apart from traditional ARQ, and investigate how to optimize the use of these resources to improve retransmission efficiency.

2) We show that for PNC atoms, non-coupled ARQ is more efficient than coupled ARQ, regardless of the error probabilities of wireless links.

To answer the second set of questions as set out at the beginning of this section:

3) We construct a Markov model to derive an idealized throughput equation that relates channel erasure probabilities with throughput. This idealized throughput is an upper bound for the throughputs of all practical ARQ schemes and it serves as a benchmark for gauging the optimality of specific ARQ schemes.

4) We put forth an ARQ design for PNC based on the standard Selective Repeat ARQ (where out-of-sequence packets are stored at the receiver) with Selective Acknowledgements (SACK). New elements unique to PNC (e.g., stored-packet tracking, non-coupled ARQ, etc.) are incorporated into the ARQ design. We show that, with proper optimization of ARQ parameters (i.e., window size and ACK frequency), our ARQ design can approach the aforementioned throughput upper bound in 3), even for actual systems in which ACKs are not error free and ACK overhead is not negligible.
We remark that in our paper, for concreteness and for a focus, the above findings are mainly introduced and studied based on the cross atom. Our findings, however, are applicable to other PNC atoms, as they are related to general principles rather than specificity of the PNC atom structure. Section VII discusses the general applicability of our proposed schemes and our findings to other PNC atoms. Furthermore, the same section explains that these design principles can also be applied when “straightforward network coding” (SNC)\(^1\) rather than PNC is used.

The rest of this paper is organized as follows. Section II overviews related work. Section III presents our system model. Section IV analyzes the benefit of storing and tracking coded and overhead packets and constructs a Markov model to derive the idealized throughput. Section V shows that non-coupled ARQ is more efficient than coupled ARQ by comparing their idealized throughputs. Section VI investigates the design of Selective Repeat ARQ with SACK in PNC. Section VII discusses the general applicability of our design principles. Section VIII concludes this paper. Due to space limit, some proofs are omitted in this paper. The proofs can be found in our technical report [3].

II. RELATED WORK

Although there has been no past work studying the ARQ design for PNC with overhead information, there has been work [4-6] on the ARQ design for SNC with overhead information.

In 2006, Katti et al. proposed COPE [4], a network relaying structure employing wireless network coding and opportunistic listening, i.e., overhearing. COPE uses SNC. Going back to the cross atom, by COPE, nodes A and B will first separately transmit A and B to the relay R in different timeslots. Then relay R will decide to transmit the coded packet \(A \oplus B\) or the native packet A or B depending on the reports from nodes C and D on the packets they have overheard. One main concern is how the coding node (relay) can be efficiently informed of the packets available at the receivers (i.e., consideration of the reporting overhead). In [5], G.S. Paschos et al. studied a deterministic system, where the contents of the receivers are announced to the coding node via reports, and a stochastic system, where the coding node makes stochastic control decisions based on statistics without explicit reports. An ARQ scheme using NACK was proposed in [6]. In the context of [5], the scheme in [6] is basically a deterministic system. Unlike [4] and [5], where the reports from a receiver are sent to the coding node periodically, in [6] a report (NACK) is triggered by the event that a coded packet has been received and yet the receiver cannot extract its desired packet out of the coded packet.

\(^1\)SNC performs network coding at the higher layer. For example, applying SNC to the same cross atom, nodes A and B should separately send their respective packet to R in two timeslots; in the third timeslot, R directly XORs packet A with B, then broadcasts \(A \oplus B\).
The ARQ studies for network coding without exploiting overheard information were proposed in [7-9]. In particular, random linear network coding is employed [7]. The sender transmits multiple random linear combinations of all packets currently in the buffer for broadcast purposes without knowing the stored packets at the receiver. Ref. [9] only studies ARQ for TWRC.

Notably, the existing ARQ studies [4-8] only focused single-hop (link-by-link) multicast transmission. In the context of the cross atom, for example, they investigated the ARQ transmissions between the relay R and the destinations C and D. After the relay multicasts coded packet to the destinations, if the destination fails to extract a desired packet due to missing overheard packet, it sends back an ACK to request the relay (which is one-hop away) to directly retransmit the desired native packet (by SNC, the relay already has native packets $A$ and $B$). The reason for directly sending native packet rather than the network-coded packet is that there is no chance for the destinations to re-overhear the overheard packets: in link-by-link ARQ, as far as the source is concerned, as soon as its packet reaches the relay correctly and is ACKed by the relay, it will not retransmit the packet again, thus overhearing of this packet is not possible. As a result, there is no need to store and track the coded packets that cannot be decoded in the link-by-link design.

Our PNC ARQ is different: we adopt the end-to-end design for multiple multi-hop unicast flows. For the cross atom, when a destination node cannot extract its desired packet, it sends an ACK to request its source node (which is two hop away) to retransmit the packet. A side benefit, besides satisfying the request from the targeted destination, is that the packet retransmissions from source nodes allow other destinations nodes to re-overhear the missed overheard packets. That is also why we need tracking. Another reason for an end-to-end design is that the relay in PNC does not attempt to decode and store native packets (we assume the relay only get XOR packets). Thus, retransmitting native packets from the relay using a link-by-link ARQ is not viable.

An additional notable point related to the ARQ studies in [4-7] is that they assumed error-free ACK and did not consider feedback overhead due to ACK. In practical wireless systems, ACK packets can be lost in transmission and sending ACK packets does consume time and can cause throughput degradation. From our analysis in this paper, such throughput degradation cannot be neglected. We investigate how to minimize throughput degradation by optimizing the transmission window size and ACK frequency.

Some ARQ implementation issues for PNC systems were proposed in [10] and [11]. However, they only consider PNC in TWRC.
III. SYSTEM MODEL

As shown in [2], a general relay network operated with PNC can be designed by decomposing it into a number of PNC building blocks. We refer to these building blocks as PNC atoms because they cannot be further decomposed into smaller building blocks. Broadly speaking, a PNC atom specifies how the traffic among a subset of nodes around a relay can be delivered via the relay using PNC.

![Transmission Window with size Ws](image)

**Fig. 2** An example of the transmission window.

The grey boxes represent packets that have been acknowledged. The white boxes represent packets that have not been acknowledged.

In this paper, for concreteness, we will first focus on a specific PNC building block with a “cross structure” as shown in Fig. 1. The cross structure consists of one relay, two source nodes, and two sink nodes. We choose to focus on the cross structure for three reasons: 1) It is an atom that utilizes overheard information. As explained in [2], there are two types of atoms, those that utilize overheard information, and those that do not. The ARQ design of atoms that do not utilize overheard information is similar to the traditional ARQ design. Only atoms that utilize overheard information will face new subtleties as addressed in this paper. 2) It is one of the three atoms that are most important for ensuring good performance [2]. In particular, in optimizing the transmission schedule by the method of network decomposition described in [2], these three atoms are used most often in the optimized solution; omitting the other atoms results in little throughput degradation. The other two important atoms are the TWRC atom, which exploits self-information but not overheard information; and the “Special” TWRC atom [2], which only utilizes overheard information in one flow. Since the cross atom utilizes overheard information in two flows and is more general, we can directly extend the ARQ design for the cross atom to the special TWRC atom. 3) The general principles of the ARQ design derived from the cross atom also apply to all other atoms that utilize overheard information. We discuss the general applicability of the ARQ principles in Section VII.

**Traffic Flows:** We consider multiple unicast traffic traversing the same relay. As shown in the traffic flow diagram of Fig. 1, node A has a stream of packets, \( A_i, i=1, 2, \ldots \), destined for node C; node B has a stream of packets, \( B_i, i=1, 2, \ldots \), destined for node D. For each of the two flows, the destination (e.g., node C) is outside the transmission range of the source (e.g., node A). The flows rely on relay R to assist the delivery packets to the destinations. As indicated in Fig. 1, the destination of each flow is within the transmission range of the source of the other flow, and can
therefore overhear the transmission that source. Specifically, node C can overhear node B, and node D can overhear node A.

**Transmission Window:** To ensure in-order delivery of packets, we set a "window" on each source. The window covers a sequence of $W_s$ packets that can be transmitted by the source. All packets to the left of the left boundary of the window have been successfully delivered to the destination. The source knows this through the ACKs fed back from the destination. The left boundary marks a packet that has not been acknowledged by the destination. All the packets to the left of the left boundary have been successfully received by the destination. The sequence number of the left-boundary packet is $SN_{\text{min}}$, as shown in Fig. 2. The right boundary contains the “largest packet” (packet with the largest sequence number $SN_{\text{min}} + W_s - 1$) that could have been sent by the source at the moment in time. The source cannot send packets to the right of the right boundary without first receiving an ACK indicating that packet $SN_{\text{min}}$ has been received at the destination (this acknowledgement will advance the left boundary and the right boundary of the window, hence allowing more packets to the right to be sent). In general, the window may contain (i) some packets that have been sent and acknowledged to be received correctly (note that packet $SN_{\text{min}}$ must not belong to the group of packets that have been acknowledged), (ii) some packets that have been sent and not yet acknowledged, and (iii) some packets that have not been sent but may be sent shortly by the source. The source sends the packets in the window in the order of their sequence numbers. When it reaches the right boundary of the window, it wraps back to the left bound of the window and retransmits $SN_{\text{min}}$. Only unacknowledged packets within a window will be retransmitted.

**Link Success Probability:** In an error-prone network, packet transmissions on a link may fail. We define link success probability ($LSP$) as the expected number of correctly received packets divided by the total number of transmitted packets—coded as well as native packets—on a link. In other words, $(1-LSP)$ is the packet loss probability on a link. In PNC systems, there are two types of links: direct link (e.g., link R-D in Fig.1) where the receiver is the destination of the transmitted packets; and overhearing link (e.g., link A-D in Fig.1) where the receiver is not the destination of the transmitted packets. Note that the link from which the relay gets the XORed packet from the superimposed signals from the sources is also counted as a direct link (its LSP is denoted as $p_1$ in Fig.1).

**Transmission pattern:** A transmission pattern is a sequence of successive transmissions by nodes in the building block to deliver one packet of each traffic flow to its destination. As shown in Fig.1, the transmission pattern of the cross structure consists of two time slots. In each time slot,
some nodes transmit and some nodes receive. The packets transmitted by the transmitting nodes and the packets received by the receiving nodes are specified in the transmission pattern in Fig. 1.

**Round:** In a PNC atom, if the flows want to deliver a stream of packets, the sequence of transmissions in its transmission pattern will be repeatedly scheduled. Each sequence of transmissions in the transmission pattern is called a **round**. Since a transmission pattern always starts with some source nodes transmitting their packets to the relay, and each source node always transmits once in a transmission pattern, each time the source nodes begin to transmit a new packet or retransmit an old packet, we say that a new **round** begins. For example in Fig.1, each time nodes A and B transmit to relay R, they start a new round. If the network is error-free, after one round, each flow successfully delivers one packet. If the network is not error-free, multiple rounds may be needed for the delivery of one packet. This paper focuses on error-prone networks.

**Coupled ARQ:** In an atom, its multiple unicast flows will not deliver their next packets until all the flows have successfully delivered their current packets. In a round, if any flow fails to deliver its current packet, the transmission pattern will repeat for all current packets of all sources in the next round. For example, with respect to the cross atom, when node A is retransmitting $A_i$ (indexed by $i$), node B must be transmitting $B_i$ (indexed by the same $i$), even if $B_i$ has been received by node D in the previous round. When all packets in the current round are delivered successfully, the multiple flows will then begin to transmit their next packets in a new round. However, there is still an outstanding subtlety. Specifically, it is possible for node A to receive ACKs indicating that both $A_i$ and $B_i$ have been received by their destinations, while node B either did not receive both ACKs or did not receive one of the ACKs. In this case, node A may progress to transmit packet $A_{i+1}$ while node B will still transmit packet $B_i$ in the next round. This causes decoupling of the transmission. In this paper, we make an idealized assumption that somehow nodes A and B know what ACKs each other receive (e.g., through a backend network or by other means). In the above example, since node B also knows the ACKs of node A, node B will also progress to transmit packet $B_{i+1}$ in the next round. We will show that despite the advantage given to coupled ARQ with this idealized assumption, it still does not perform as well as non-coupled ARQ.

**Non-coupled ARQ:** Each unicast flow will deliver its next packet after its current packet has been successfully extracted at its destination, regardless of the outcomes at other flows.

**End-to-end ARQ:** This paper considers end-to-end ARQ designs. In end-to-end ARQ, the relay does not participate in the ARQ process. Specifically, the feedback (ACK) is provided by destination nodes only. The relay only forwards the ACK from the destination nodes to the source
nodes. In addition, only the source nodes retransmit packets; the relay does not store packets for retransmission purposes.

IV. STORED-PACKET TRACKING AND THROUGHPUT ANALYSIS

We explained the importance of store-packet tracking by means of an example in Section I. Specifically, the destination node should track coded packets and overheard packets, so that it can extract its desired packets from coded packets and overheard packets received from different round. This section formally presents this tracking design and constructs a Markov model to analyze the resulting throughput.

Recall that we also defined coupled ARQ and non-coupled ARQ in Section III. Since the tracking design for non-coupled ARQ is more intricate, and non-coupled ARQ is more efficient than coupled ARQ (to be shown in Section V), we focus on the stored-packet tracking design for non-coupled ARQ here.

A. Stored-Packet Tracking

For PNC systems with overheard information, such as the cross structure under study, successful extraction of a packet at the destination depends on the availability of both the coded packet and an associated overheard packet. It is possible that, in some rounds, the destination node only receives either the coded packet or the overheard packet. Although the packet extraction will fail for that round, the received packet can be stored for future use. This is the case for both coupled and non-coupled ARQs. In our scheme, the destination nodes separately store the overheard packets and the coded packets in an overheard-packet pool (O-pool) and a coded-packet pool (C-pool), respectively.

Since each native packet can be retransmitted by the source multiple times before it is successfully extracted at the destination, and different flows are not coupled, a packet $A_i$ that is retransmitted in a round may be network-coded with a packet $B_j$ where the index $j$ may not be the same as $i$. In particular, an overheard packet may be used to extract desired packets from different coded packets. For example, it is possible for node D to receive $A_i \oplus B_i$, $A_i \oplus B_j$ and $A_i \oplus B_k$ in three different rounds. These three coded packets contain the same $A_i$, thus if node D overhears $A_i$ in some round, it can then successfully extract $B_i$, $B_j$ and $B_k$ from the three coded packets. In other words, if in a round, the destination node only receives a coded packet but fails to receive the overheard packet, the coded packet may still be decoded by a side packet overheard in a past or future round. Proper stored-packet tracking requires that each time the destination overhears a packet (e.g., $A_i$), it checks all the stored coded packets (e.g., $A_i \oplus B_i$, $A_i \oplus B_j$ and $A_i \oplus B_k$) in C-pool to see whether some desired packets can be decoded using these packets and the overheard in-
formation just received. By the same token, each time the destination receives a coded packet (e.g., \(A_i \oplus B_k\)), if it fails to overhear the corresponding packet in the same round (e.g., \(A_i\)), it should track the stored overheard packets (e.g., \(A_i\)) in *O-pool* to see whether an overheard packet can be used together with the coded packet to extract a desired packet.

Note that since an overheard packet (e.g., packet \(A_i\) received by node D in the above example) can be used to decode multiple coded packets, even if the extraction of the desired packet is successful in the round when the overheard packet is received (e.g., node D also receives the coded packet \(A_i \oplus B_i\) so that node D can extract packet \(B_i\)), it will still be useful to store the overheard packet (e.g., for node D to store \(A_i\)) for potential future use. This is because the target destination of the overheard packet may not have received the packet in this round, and that the source of the overheard packet will need to retransmit the overheard packet (e.g., node A retransmits \(A_i\) because node C did not receive it in this round). When the source retransmits the packet, it may be network-coded with yet another packet by the relay (e.g., while node A retransmits \(A_i\), node B transmits \(B_j\) in a future round so that the relay forwards \(A_i \oplus B_j\)). The stored overheard packet can then be used to extract a desired future packet even if this future retransmission fails to be overheard (e.g., node D cannot receive \(A_i\) in this future retransmission, but node D already has \(A_i\) stored from a previous round; node D can then use the stored \(A_i\) and the received \(A_i \oplus B_j\) to extract \(B_j\)).

Another issue of stored-packet tracking then arises as to when a stored packet can be removed from *O-pool* or *C-pool*. Before a packet is removed, we need to ascertain that it will be no longer useful in the future. Let us first consider the rule for removing packets from the *C-pool*. In general, a coded packet in *C-pool* can be removed if the desired packet embedded in it has been successfully extracted (e.g., node D would drop \(A_i \oplus B_i \forall x\) if \(B_i\) has been extracted). This is because the coded packet will no longer be useful for the extraction of the desired packet given that the desired packet has been obtained. Now, let us consider the *O-pool*. In general, an overheard packet in the *O-pool* can be removed as soon as the packet that has been successfully delivered by its original flow (e.g., packet \(A_i\) can be dropped as soon as it is successfully delivered to node C). This is because the packet will not be transmitted by its source node anymore and hence no future XOR packet will embed the packet (e.g., no future XOR packet relayed by the relay will be of the form \(A_i \oplus B_i \forall x\)).

Overall, *stored-packet tracking* is a mechanism that not only keeps track of which stored packets can be used for extraction, but also keeps track of which stored packets can be discarded because they are no longer useful. In the next section, we investigate the extent to which tracking can increase throughput.
B. Throughput Markov model for non-coupled ARQ with window size 1

In general, the throughput analysis of an ARQ system with tracking can be rather complex. Without packet storage and tracking, the throughput of the cross atom is \( p_1(p_2p_4 + p_3p_5) \) packets per round, where \( p_i \) are the LSPs (link successful probabilities) on the respective links as indicated in Fig. 1.

With packet tracking, the throughput is not immediately apparent. To study the throughput with stored-packet tracking, here we provide a closed-form analysis for an idealized ARQ model. The throughput of this idealized ARQ model serves as an upper bound for the throughputs of more realistic ARQ models. Later in Section VII, we show by simulations that the throughput of the realistic ARQ can be very close to the throughput of the idealized ARQ.

We construct a Markov model to analyze the throughput of the idealized ARQ model. The following assumptions are made in the idealized ARQ: (1) immediate error-free ACK feedback, (2) negligible ACK transmission time. With the ideal model, each source immediately knows whether the current packet has been successfully delivered.

To reduce the number of states for analytical tractability, we also assume that the transmission window size of each flow is one, that is, the source node will keep on sending the same packet until this packet has been successfully delivered. Intuitively, this assumption of window size being one under the ideal model will not affect the decoding probability, hence the throughput. For example, in the cross PNC atom, suppose that node D overhears packet \( A_i \) but \( A_i \) is not successfully delivered to node C. Thus, node A will retransmit \( A_i \) in a future round (if the window size is one, then it is in the next round; if the window size is larger than one, the retransmission may not happen immediately in the next round). In a future round, if node D receives a packet \( A_i \oplus B_j \), it can extract \( B_j \) by exploiting the stored \( A_i \). Note that \( B_j \) can be any packet in the window. For larger window, there are more possible choices of packet \( B_j \) that could be transmitted simultaneously with packet \( A_i \). However, which packet \( B_j \) is paired with \( A_i \) does not affect the probability of receiving packet \( A_i \oplus B_j \) at the destination nodes. Thus, in that sense, having a window size larger than one will not improve the probability of delivering a coded packet. Neither will the probabilities of overhearing packets \( A_i \) and \( B_j \) be improved. For another example, suppose that node D receives \( A_i \oplus B_j \) but misses the overheard packet \( A_i \). So long as \( A_i \) is not successfully delivered to node C (hence \( A_i \) should be retransmitted), node D always has the same probability to re-overhear \( A_i \) in some future round, hence can decode the stored \( A_i \oplus B_j \). The larger window size just means \( A_i \) would be retransmitted later but does not improve the overhearing probability of \( A_i \).
To validate our intuition, we have performed massive simulations based on different window sizes. The results show that their throughputs are the same regardless of window size\(^2\).

For simplicity, we also assume in our analysis that the LSP of different direct links are homogeneous (denoted by \(p_1\)) and the LSP of different overhearing links are also homogeneous (denoted by \(p_2\)). As far as the general principles for PNC ARQ is concerned, we find that the conclusions obtained from analysis of the homogeneous study in this paper also apply to the inhomogeneous case generally. Although a rigorous proof is not available due to the analytical complexity of the inhomogeneous case generally, these findings have been validated by our simulation results.

1. Two Throughput States:

Since the network is error-prone, there may be some rounds with no successful delivery to either node C or node D. For rounds with successful deliveries, there are two possibilities: 1) only one of the two flows successfully delivers its packet; 2) both flows successfully deliver their packets. Accordingly, we first construct a “macroscopic” Markov chain with two states only (see Fig. 3): **State 1**, the system just delivered one packet (the last successful round delivered one packet); **State 2**, the system just delivered two packets (the last successful round delivered two packets).

The transitions in the Markov chain may take varying amounts of time, depending on how many unsuccessful rounds there are before the next successful round. Each transition in Fig. 3 may take several rounds until at least one desired native packet is successfully decoded at one of the destinations.

We refer to the two macro states in Fig. 3 as **throughput states**. The determination of the probabilities of the throughput states, and the average amounts of time the systems stays in the states before the next transition, give us the system throughout, as explained below. With reference to Fig. 3, let \(P_s\) be the equilibrium distribution of state \(S, S \in \{1,2\}\). Let \(T_s\) be the average sojourn time in state \(S\) before the next transition, measured in number of rounds. A transition is triggered when either one or two packets have been delivered in a round. Note that upon a transition, the next state can be either the same state or the other state. The system throughput (in number of packets per round) is

\[2 \cdot \frac{p_{11} + p_{22}}{1 - (p_{11} + p_{12} + p_{21} + p_{22})} \]

\(^2\) We omit the results here to conserve space.
\[ T_h = \frac{P_1 + 2P_2}{P_1 \cdot T_1 + P_2 \cdot T_2} \] (1)

2. Calculating the equilibrium distributions \( P_1 \) and \( P_2 \):

First, we have \( P_1 + P_2 = 1 \). Second, with respect to the Markov chain in Fig. 3, we have the balance equation: \( P_1 \cdot p_{12} = P_2 \cdot p_{21} \), where \( p_{ij} \) denote the transition probability from macro state \( i \) to macro state \( j \).

Next, we need to derive \( p_{12} \) and \( p_{21} \). We can model a macroscopic transition in Fig. 3 with a transient “microscopic” Markov process, as described below. A microscopic Markov process with some initial state is launched when a macroscopic transition occurs. The microscopic process then goes through a sequence of microscopic state transitions until it reaches an absorbing state, which triggers the next macroscopic transition. This kicks off another transient microscopic Markov process. Whereas each macroscopic transition may take several rounds, each microscopic transition takes exactly one round.

We first list the possible microscopic states of the microscopic Markov process (note that with window size of one, at a destination, at most one coded packet and at most one overheard packet will be stored): State \( \phi \): Both destinations do not have a coded packet or an overheard packet.

State O: One destination has an overheard packet but no coded packet; the other destination has neither.

State X: One destination has a coded packet but no overheard packet; the other destination has neither.

State XO: One destination has a coded packet but no overheard packet; the other destination has an overheard packet but no coded packet.

State OO: Both destinations have an overheard packet but no coded packet.

State XX: Both destinations have a coded packet but no overheard packet.

Upon a macroscopic transition that lands the system in state 2 in Fig. 3, the system will begin at state \( \phi \) at the microscopic level in Fig. 4. This is because both source nodes will begin transmitting new native packets and that all stored information at the destinations is no longer useful. On the other hand, upon entering state 1 in Fig. 3, the system will begin at state O at the microscopic level in Fig. 4, as explained below. When only one native packet is successfully decoded (e.g., packet \( A_1 \) is decoded at its destination node C, but packet \( B_1 \) is not yet decoded at its destination node D), the successful destination (e.g., node C) must have an overheard packet (e.g., packet \( B_1 \)) from the unsuccessful source node (e.g., node B). Since the unsuccessful source node (e.g., node B) will still need to retransmit the same packet (e.g., \( B_1 \)) in the next round, and the successful destina-
tion (e.g., node C) already has already overheard that packet (e.g., \(B_1\)), for the next round, the successful destination only need to receive the new coded packet (e.g., \(A_2 \oplus B_1\)) and will not need to receive the overheard packet again (e.g., \(B_1\)). Meanwhile, for the unsuccessful destination (e.g., node D), even if it already has an overheard packet from the current round (e.g., \(A_1\)), this overheard packet can no longer be used for decoding the next coded packet (e.g., \(A_2 \oplus B_1\)); similarly, even if the unsuccessful destination (e.g., node D) already has the coded packet from the current round (e.g., \(A_1 \oplus B_1\)), this stored coded packet is no longer useful in the next round. Hence, at the beginning of the next round, the storage of the other destination has neither a useful coded packet nor a useful overheard packet.

As depicted in Fig. 4, at the microscopic level, the system may visit other microscopic states before the next macroscopic transition is triggered. For example, the system could begin at state O (microscopic), then go to state XO (microscopic), then go to state 2 (macroscopic). In Fig. 4, states 1 and 2 are the absorbing states corresponding to where the system lands at the macroscopic level. We break the graph into several subgraphs to avoid cluttering. It should be understood that these subgraphs should be pieced together for the overall transient Markov chain. For example, state O in the first subgraph is the same state O of the second subgraph.

Each transition in Fig. 4 takes exactly one round. The transition probabilities in the graphs in Fig. 4 can be easily calculated. For example, \(P_{X \rightarrow XO}\) represents the transition probability from micro state X to micro state XO, that is, the destination with no useful stored packet acquires an overheard packet (with probability \(p_2\)) but misses the coded packet (with probability \((1-p_1^2)\)), and the destination that already has the coded packet fails to overhear the required packet (with probability \((1-p_2)\)). So \(P_{X \rightarrow XO} = p_2 (1-p_1^2) (1-p_2)\).
In Fig. 4, the transient Markov chain will end in one of the absorbing states corresponding to the macro throughput state $S, S \in \{1,2\}$. Let $p_{i}^{(S)}$ denote the probability that system will end in absorbing state $S$ given that it is now in microscopic state $i$. Since $S \in \{1,2\}$, $p_{i}^{(0)} + p_{i}^{(2)} = 1 \ \forall i \in \{1,2\}$ in Fig. 4. Then $p_{i}^{(S)}$ can be calculated by $p_{i}^{(S)} = \sum_{j} p_{i,j}^{(S)} = \frac{1}{1 - p_{i,-i}} \sum_{j \neq i} p_{i,j}^{(S)}$. Moreover, since upon entering macroscopic states 1 and 2, the system will begin at microscopic states $O$ and $\phi$, we can see that $p_{12} = p_{o}^{(S)}$ and $p_{21} = p_{\phi}^{(S)}$. Remembering that $P_{1} + P_{2} = 1$ and $P_{1} \cdot P_{2} = P_{2} \cdot P_{21}$ (as shown in Fig. 3), we can get $P_{1}$ and $P_{2}$ by $P_{1} = \frac{P_{21}}{P_{21} + P_{21}}$ and $P_{2} = \frac{P_{12}}{P_{12} + P_{21}}$, where $P_{1}$ and $P_{2}$ are the macro-state stationary probabilities.

3. Calculating the expected transmission time $T_{1}$ and $T_{2}$:

As mentioned before, not every round leads to a successful delivery (i.e., triggers a new throughput state), after entering a new macro throughput state, it could take several unsuccessful rounds before the transition to the next macro throughput state. To get $T_{1}$ and $T_{2}$, we also turn to the microscopic Markov process (in Fig.4) to calculate the expected number of transitions (rounds) starting from an initial state (1 or 2) until an absorbing state (1 or 2).

Let $E[T | i]$ denote the expected number transitions until the system reaches an absorbing state given that the current microscopic state is state $i$. We can write:

$$E[T | i] = p_{i,-i} \cdot (E[T | i] + 1) + \sum_{j \neq i} p_{i,j} \cdot (E[T | j] + 1) = \frac{1}{1 - p_{i,-i}} \left( \sum_{j \neq i} (p_{i,j} \cdot E[T | j]) + 1 \right).$$

The detailed equations for different microscopic states $i$ can be derived from the transition graphs in Fig. 4. For example, $E[T | O] = \frac{1}{1 - p_{o,o}} (p_{o,o} \cdot E[T | O] + p_{o,o} \cdot E[T | o] + 1)$ and $E[T | o] = \frac{1}{1 - p_{o,o}}$. Moreover, since upon entering macroscopic states 1 and 2, the system will begin at microscopic states $O$ and $\phi$, we can see that $T_{1} = E[T | O]$ and $T_{2} = E[T | \phi]$. By solving a set of linear different equations, we can get $T_{1}$ and $T_{2}$.

Substituting all the solved variables into (1), we get a rather complicated expression:

$$T_{1} = \frac{(2p_{1}^{2}(1-p_{2})^{2} - (-2 + p_{2})^{2})^{2} - (6 - 2p_{2} + p_{2}^{2})^{2} - (1 + p_{2})^{2}(-1 + p_{2})^{2}(-12 + 17p_{2} - 11p_{2}^{2} + 4p_{2}^{3}) - p_{2}^{2}(-1 + p_{2})^{4}(-8 - 3p_{2} - 2p_{2}^{2} + 2p_{2}^{3}) - p_{2}^{2}(-1 + p_{2})^{4}(-44 + 76p_{2} - 43p_{2}^{2} + 10p_{2}^{3}) + p_{2}^{2}(-1 + p_{2})^{4}p_{2}(-36 + 57p_{2} - 39p_{2}^{2} + 11p_{2}^{3}) + 8p_{2}^{2}(-48 + 84p_{2} - 45p_{2}^{2} + 8p_{2}^{3}) - p_{2}^{4}(-1 + p_{2})^{4}p_{2}(-56 + 102p_{2} - 62p_{2}^{2} + 13p_{2}^{3}) + p_{2}^{2}(-16 + 42p_{2} - 39p_{2}^{2} + 15p_{2}^{3} - 2p_{2}^{4}) + 3p_{2}^{2}(-2 + p_{2})^{2}p_{2}^{2}(3 - 5p_{2} + 2p_{2}^{2})}{(2p_{1}^{4}(-1 + p_{2})^{2}p_{2}^{2} - p_{1}^{13}(-1 + p_{2})^{3}(1 + 10p_{2} + p_{2}^{2}) + p_{1}^{12}(-1 + p_{2})^{2}(3 + 20p_{2} + 3p_{2}^{2}) - p_{1}^{11}(-1 + p_{2})^{2}p_{2}(20 + p_{2}) - p_{1}^{10}(-1 + p_{2})^{2}(13 - 22p_{2} + 17p_{2}^{2} - 6p_{2}^{3} + p_{2}^{4}) + p_{1}^{9}(-1 + p_{2})^{2}(-24 + 64p_{2} - 93p_{2}^{2} + 77p_{2}^{3} - 31p_{2}^{4} + 5p_{2}^{5}) - p_{1}^{8}(-1 + p_{2})^{2}(-16 + 29p_{2} - 41p_{2}^{2} + 42p_{2}^{3} - 25p_{2}^{4} + 5p_{2}^{5}) - p_{1}^{7}(-1 + p_{2})^{2}p_{2}(66 - 161p_{2} + 147p_{2}^{2} - 57p_{2}^{3} + 9p_{2}^{4}) + p_{1}^{6}(-1 + p_{2})^{2}p_{2}(60 - 164p_{2} + 167p_{2}^{2} - 79p_{2}^{3} + 15p_{2}^{4}) + \ldots}$$
The above throughput formula is tedious to obtain by hand computation. We used a symbolic computation tool (e.g., Mathematica) to calculate it.

|          | $P=0.9$ | $P=0.85$ | $P=0.8$ | $P=0.75$ | $P=0.7$ | $P=0.65$ | $P=0.6$ | $P=0.55$ | $P=0.5$ |
|----------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| PNC-T ($p/r$) | 1.45    | 1.22    | 1.02    | 0.84    | 0.68    | 0.54    | 0.43    | 0.33    | 0.25    |
| PNC-NT ($p/r$) | 1.50    | 1.31    | 1.14    | 0.99    | 0.85    | 0.73    | 0.62    | 0.53    | 0.44    |
| Gain      | 3%      | 6%      | 11%     | 17%     | 24%     | 33%     | 44%     | 58%     | 75%     |

$p/r$: packets per round.

C. Performance Gain of Stored-Packet Tracking and More Discussion on Tracking

In Part B, because of the idealized assumption of error-free and overhead-free ACKs, using window size of one is good enough. Using window size of one, in turn, allows a simple tracking method in which at most one coded packet and at most one overheard packet need to be stored. Nevertheless, this simple tracking can already give rise to significant throughput improvement. The resulting throughput, $T_{h1}$, obtained under the ideal assumptions of error-free and overhead-free ACK, serves an upper bound for the practical ARQ schemes where these ideal assumptions are dropped, which will be discussed later. In other words, we will measure our practical ARQ schemes using $T_{h1}$ as the benchmark.

By adopting stored-packet tracking under non-coupled ARQ as analyzed in Part B, higher throughput can be obtained than without packet storage and tracking. In Table 1, we show the throughput gains of PNC with tracking (PNC-T) relative to conventional PNC without tracking (PNC-NT) under different LSPs (assuming homogenous link successful probability for all direct and overhear links). The throughput gain is larger when the wireless channel becomes worse. As shown, the throughput gains increase from 3% to 75% when the LSP decreases from 0.9 to 0.5. Note that we do not list the results when $p$ is too small, because when $p$ is too small, we would be better off using the traditional hop-by-hop transmission scheme. According to the benchmark throughput formulation of PNC, to ensure that PNC is more efficient than traditional hop-by-hop transmission whose throughput is $p/2$, $p$ must be larger than 0.57.

D. More Discussion on Stored-Packet Tracking

The tracking described so far is a single-iteration tracking mechanism. Each time the destination receives an overheard packet (or a coded packet), it checks the stored coded packets in the C-pool (or overheard packets in the O-pool) to see whether a desired native packet can be extracted by combining received packet with a stored packet. However, under window size larger than one, tracking can go further, using multi-iteration tracking. For multi-iteration tracking, after an itera-
tion in which a native packet is extracted, it proceeds to another iteration to check if the newly extracted native packet can be combined with a stored packet in the C-pool to extract another packet. This process repeats itself until no more packet can be extracted\(^3\). For example, suppose that node D has \(A_1 \oplus B_1\), \(A_1 \oplus B_2\), and \(A_2 \oplus B_2\) in its C-pool. When node D overhears \(A_2\), it can extract \(B_2\) by \(A_2 \oplus (A_2 \oplus B_2)\). In the next iteration, it combines \(B_2\) with \(A_1 \oplus B_2\) to extract \(A_1\). In the iteration after that, node D combines \(B_1\) with \(A_1 \oplus B_1\) to decode \(A_1\). Such multi-iteration tracking/decoding can improve throughput.

However, from above example, we can see that only when certain particular combinations of coded packets are stored can multi-iteration tracking/decoding be useful. Our simulation experiments indicate that the desired native packets obtained from multi-iteration tracking (tracking beyond the first iteration) constitute only a small portion of the overall desired native packets extracted (less than 3\% for different window size, inhomogeneous link successful probability and all PNC atoms). Thus, there is little need to include multi-iteration tracking in our design. In this paper, the single term tracking always represents single-iteration tracking.

V. COUPLED ARQ VERSUS NON-COUPLED ARQ

In this section, we compare non-coupled ARQ with coupled ARQ through their throughputs analyses. We have given the throughput of non-coupled ARQ in last section by constructing Markov model. For coupled ARQ, we can also calculate its throughput using a similar model and adopting same assumptions (window size 1, homogeneous LSP, etc.) in Section IV. B. An intuitive comparison proof for inhomogeneous LSP is given in Section V.C.

A. Throughput of Coupled ARQ with window size 1

In coupled ARQ, even if one source node (e.g., node A) already knows that the currently transmitted native packet (e.g., packet \(A_1\)) has been successfully extracted at its destination node (i.e., node C), it will still transmit this packet (e.g., packet \(A_1\)) until the other source node (e.g., node B) also successfully delivers its currently transmitted native packet (e.g., packet \(B_1\)) to its destination node (i.e., node D). After both sources have delivered the current packets, they move on to transmit their next packets (e.g., packets \(A_2\) and \(B_2\)). That is to say, for coupled ARQ, we can only consider one macro throughput state: both flows successfully deliver their packets. Hence, to calculate the throughput of coupled ARQ, we do not need to construct a macroscopic Markov chain as the one shown in Fig. 3. We just directly compute the expected number of

\(^3\) This process is similar to the belief-propagation decoding of LDPC erasure codes.
rounds (denoted by $E[T_n]$) from the initial state (both sources begin to transmit two new packets, i.e., microscopic state $\phi$) to the ending state (these two packets have been successfully delivered, i.e., throughput state 2). The computation adopts the same method as in non-coupled ARQ. However, compared with non-coupled ARQ, there are additional three microscopic states that need to be tracked (Due to space limit, we do not draw transition graphs for coupled ARQ, as they are similar to those transition graphs for non-coupled ARQ in Fig. 4): 

State 1$\phi$: One destination already has the desired native packet; the other destination has nothing (neither the coded packet nor the overheard packet).

State 1O: One destination already has the desired native packet; the other destination has an overheard packet but no coded packet.

State 1X: One destination already has the desired native packet; the other destination has a coded packet but no overheard packet.

Then, we can get $E[T_N]$ by

$$E[T_N] = p_{\phi\rightarrow\phi} \cdot (E[T_N] + 1) + \sum_{i=1}^{n} p_{\phi\rightarrow i} \cdot (E[T_N | i] + 1) = \frac{1}{1 - p_{\phi\rightarrow\phi}} \left( \sum_{i=1}^{n} (p_{\phi\rightarrow i} \cdot E[T_N | i]) + 1 \right),$$

where $E[T_N | i]$ denotes the expected transition rounds from microscopic state $i$ to the ending throughput state (the currently transmitting two packets have been successfully delivered).

Finally, by Mathematica, we can get $Th_2 = 2/E[T_N]$. (The complete throughput formula is as tedious as $Th_1$ and there is no new principle, so we do not give the complete form here to save space.)

B. Comparison of Non-Coupled ARQ and Coupled ARQ

After getting the throughputs of non-coupled ARQ ($Th_1$) and coupled ARQ ($Th_2$), let us compute their difference, $f(p_1, p_2)=Th_1 - Th_2$, which is a polynomial function of $p_1$ and $p_2$. Although it is hard to handle the function by hand computation, we can again resort to a symbolic computation tool (e.g., Mathematica). We first show that $f(p_1, p_2)>0$ at some points $p_1$ and $p_2$. $0< p_1<1$ and $0< p_2<1$. Then we use Mathematica to show the polynomial equation $f(p_1, p_2)=0$ has no solution when $0< p_1<1$ and $0< p_2<1$. This means $f(p_1, p_2)\neq 0$ within this interval. Since $f(p_1, p_2)$ is a continuous function of $p_1$ and $p_2$ (as polynomial function is continuous), $f(p_1, p_2)>0$ (i.e., $Th_1 > Th_2$) where $0< p_1<1$ and $0< p_2<1$. So we can say that non-coupled ARQ is more efficient than coupled ARQ with window size 1. Therefore we focus on non-coupled ARQ in PNC ARQ design.

---

4 Mathematica can judge whether the polynomial equation has no solution by the Cylindrical Algebraic Decomposition (CAD). Please refer to: http://mathworld.wolfram.com/CylindricalAlgebraicDecomposition.html
C. Intuitive Inhomogeneous Probability Proof

The coupled ARQ and the non-coupled ARQ make differences (contribute different throughputs) only when the coupled system transits to one its three microscopic states: State 1φ, State 1O and State 1X as mentioned in section A, and the non-coupled system transits to State 1. We compare the two schemes under the three microscopic states of the coupled system separately to see which scheme is more efficient. Without loss of generality, we assume node A has just successfully delivered a packet \( A_1 \) to node C; node B has just failed to deliver a packet \( B_1 \) to node D. Going forward, node B would still transmit packet \( B_1 \) for both coupled and non-coupled systems. In the non-coupled system, node A would progress to transmit packet \( A_2 \), while in the coupled system node A would still transmit packet \( A_1 \). In the following, we go through the three possible microscopic states in the coupled system and compare the corresponding situations going forward in the coupled and non-coupled systems:

State 1φ: For the non-coupled system, node A has a chance to deliver a new packet \( A_2 \). However, the tradeoff is that node D will not be able use the stored overheard packet \( A_1 \) or coded packet \( A_1 \oplus B_1 \) to decode its desired packet \( B_1 \) from node B. In other words, the non-coupled system has no useful stored information. Now, in State 1φ, node D in the coupled system has no useful stored information either. Thus, allowing node A to progress to the transmission of packet \( A_2 \) in the non-coupled system will lead to higher efficiency, since node C will have a chance to obtain a new desired packet.

State 1O: in this state, node D already has the overheard packet \( A_1 \). In the next round, for the non-coupled system, since node A transmits \( A_2 \), the stored packet \( A_1 \) in node D is not useful for the decoding of packet \( B_1 \). For the non-coupled system, the possible throughput in the next round is \( p_1(p_2p_4+p_5) \) packets, where \( p_i \) are the success probabilities on the respective links as indicated in Fig. 1. In particular, relay R receives the coded packet \( A_2 \oplus B_1 \) with probability \( p_1 \); to extract packet \( B_1 \), node D needs to receive \( A_2 \oplus B_1 \) from R (with probability \( p_4 \)) and overhear \( A_2 \) from node A (with probability \( p_2 \)); to extract packet \( A_2 \), node C only needs to receive \( A_2 \oplus B_1 \) from R (with probability \( p_5 \)), because it has overheard \( B_1 \) (one of the prerequisites of successfully extracting \( A_1 \) in the last round). For the coupled system, the possible throughput (i.e., successfully delivering packet \( B_1 \)) in next round is \( p_1p_4 \) packets per round, because node D can reuse the stored packet \( A_1 \). To compare the efficiencies of two systems, let us look at \( p_1(p_2p_4+p_5) \) and \( p_1p_4 \). When
The coupled system is more efficient. However, for the system transit to State 1O frequently, intuitively $p_5$ should be larger than $p_4$ (i.e., with a relatively high probability, node C successfully receives the packet from the relay but node D fails to receive it) and $p_2$ should be larger than $p_4$ (i.e., with a relatively high probability, node C successfully overhears from node A but fails to receive the packet from the relay). When $p_5 > p_4$ and $p_2 > p_4$, $p_2p_4 + p_5 > p_4$, then non-coupled system is more efficient. Thus, intuitively, for a set-up in which the probability of landing in state 1O is non-negligible, the non-coupled system is more efficient.

State 1X: In this state, node D already has the coded packet $A_1 \oplus B_1$. In the next round, for the non-coupled system, since node A transmits $A_2$, the stored packet $A_1 \oplus B_1$ in node D is not useful for the decoding of packet $B_1$. For the non-coupled system, the possible throughput in the next round is $p_1(p_2p_4 + p_5)$ packets, as analyzed in the last paragraph. For the coupled system, the possible throughput (i.e., successfully delivering packet $B_1$) in next round is $p_2$ packets per round, because node D can reuse the stored packet $A_1 \oplus B_1$. To compare the efficiencies of two systems, let us look at $p_1(p_2p_4 + p_5)$ and $p_2$. When $p_1(p_2p_4 + p_5) < p_2$, the coupled system is more efficient. However, for the system transit to State 1X frequently, intuitively $p_1p_5$ should be larger than $p_2$ (i.e., with a relatively high probability, node C successfully receives the coded packet through the relay but node D fails to overhear from node A). When $p_1p_5 > p_2$, $p_1(p_2p_4 + p_5) > p_2$, then non-coupled system is more efficient. Thus, intuitively, for a set-up in which the probability of landing in state 1X is non-negligible, the non-coupled system is more efficient.

After going through all three possible microscopic states in the coupled system, we can say that to go forward, the non-coupled systems are always intuitively more efficient than the coupled systems.

VI. ERROR-PRONE ACK

In Section IV, to find the theoretically optimal throughput, we neglected the air time used for sending ACKs (acknowledgement) and assumed that ACKs are error-free. However, in a practical wireless system, sending ACK packets does consume time and ACK packets can be lost in transmission, leading to throughput degradation. In this section, we investigate how to design the ARQ mechanism to allow the throughput of a real system to approach the ideal benchmark throughput in Section IV despite ACK errors and transmission overhead.

A. PNC ARQ

ACK is an important function in ARQ. ACK packets are numbered according to the packets
that the receiver has received. ACKs allow the transmitter to adjust its transmission window according to the packets already received by the receiver. If the transmitter receives ACKs that indicate certain packets have been received, the transmitter will not transmit these packets again; the unACKed packets, however, will be transmitted/retransmitted in the future.

Wireless networks are error-prone and ACKs can be lost in transit. With missing ACKs, the transmitter may retransmit packets already received at the receiver (i.e., there may be a discrepancy between the reception status as perceived by the transmitter and the actual reception status at the receiver). To eliminate such wasteful packet retransmissions, we design an ARQ based on Selective Repeat (SR) (where out-of-sequence packets are stored at the receiver) with Selective Acknowledgements (SACK) [8]. Hereafter, we refer to our ARQ design, which incorporate stored-packet tracking, non-coupled ARQ and SR with SACK, as PNC ARQ.

Selective Repeat selectively retransmits packets. Specifically, with selective repeat, the transmitter sends packets within a window in the order of their sequence numbers, skipping those packets that it knows to have been received by the receiver already. When it reaches the right boundary of the window, it wraps back to the left bound of the window to retransmit packets (referred to as boundary wrap back) in the order of the sequence numbers again. Updates as to which packets have been received are performed whenever an ACK is received from the receiver.

The receiver accepts out-of-order packets. The receiver sends SACKs specifying which packets within a window have been received and which have not been received. In particular, besides the cumulative ACK indicating that all packets with a sequence number smaller than the ACK number have been received, the selective ACK also indicates the noncontiguous received packets with a sequence number larger than the cumulative ACK number. This allows the transmitter to more selectively retransmitting packets, reducing the likelihood of retransmitting packets already received.

Let us introduce the format of SACK frame in PNC ARQ with an example: \( \{7, 0100101\} \). The first number indicates that the destination node has received all packets before 7, the subsequent bit map indicates the packets after packet 7 that have been received— the length of the bit map is just \( W \) (window size) minus 1. In the example, the destination node has received packets 9, 12 and 14. The SACK frame tells the source that packets 7, 8, 10, 11, 13 are yet to be received.

With SACK, when window size is large, although some ACKs could be lost, so long as a subsequent ACK (triggered by reception of a subsequent packet) is received, the transmitter can update the information of all previously received packets. That is to say, after successfully delivering a packet, as long as the transmitter receives an ACK any time before the transmission pro-
cess reaches the right window boundary and wraps back to the left window boundary, wasteful retransmissions can be avoided.

Last but not least, we adopt an ACK-oriented wrapping back mechanism in PNC-ARQ. Specifically, after receiving an ACK and updating the packet information (window), the transmitter will immediately wrap back and retransmit the unACKed packets from the beginning of the window (referred to as ACK wrap back), because we want to advance the left boundary of the window as quickly as possible. This is different from wrapping back due to reaching the right window boundary in the last paragraph and does not entail wasteful retransmissions of packets that have been received. We define the transmissions in between two consecutive ACK wrap back as a transmission cycle. If there is a boundary wrap back within a transmission cycle, there will be wasteful retransmissions. For example, assume window size of five and the transmitter starts transmitting packets in sequence from packet 1 to packet 5. Suppose that after the transmitter transmits packet 4 and before it transmits packet 5, it receives an ACK \( \{3, 1000\} \), it will immediately wraps back to retransmit packet 3. This is an ACK wrap back. Note that the ACK \( \{3, 1000\} \) indicates that packet 4 has been received. We can deduce packet 3 has not been received. Furthermore, the ACK \( \{3, 1000\} \) contains full information on the reception status at the receiver at the moment packet 4 is received. Thus, it makes sense for the transmitter to immediately wrap back to retransmit to make up for the missing packets up to packet 4. In this example, after the ACK wrap back, the right window boundary is packet 7. Say, after this wrap back, the transmitter does not receive any ACK even after it transmits packet 7, it will wrap back to retransmit packet 3. That is a boundary wrap back. Since packet 3 may have been successfully received (but not successfully ACKed), the retransmission of packet 3 can potentially be wasteful.

\[ \text{Parameters in PNC ARQ} \]

In PNC ARQ, two parameters can affect system throughput. The first parameter is the transmission window size \( W \). As mentioned in Section A, if the transmission process reaches the right boundary of the transmission window without receiving an ACK since it last started from the left window boundary, the transmitter has to retransmit starting from the left boundary of the window. In that case, some unACKed packets in the window may have been successfully received, and the retransmissions of them will be wasteful. A larger window size gives a higher probability of receiving an ACK before right-to-left-boundary wrap back (i.e., less wasteful retransmissions). However, a larger \( W \) also means that an ACK packet should contain more bits (i.e, \( W-1 \)) to indicate the delivery state of each packet in the window. As a result, the system has to consume more air time to send back each ACK, leading to a larger feedback overhead (i.e., the mean air time consumed by ACKs relative to the mean air time consumed by Data packets). So the window size
should be bounded in a practical system. We need to find a proper $W$ to trade off between these two effects.

The second parameter is the ACK frequency, that is, the rate at which ACK is returned, expressed as the number ($N$) of Data packet receptions per ACK. Here, in our atom, one reception means a coded packet that contains a native packet destined for this node is successfully received from the relay by the destination (regardless of whether this coded packet can be decoded at the time). Transmitting an excessive number of ACKs may lead to large feedback overhead. Since each selective acknowledgment (SACK) always includes the delivery states of all previously received packets, when window size is large, there is no need to send back ACK for each reception. To reduce the feedback overhead caused by ACK in our design, the destination node will send an ACK after every $N$ successful receptions from the relay. Larger $N$ leads to smaller feedback overhead. However, given a window size $W$, increasing $N$ would increase the chance of wasteful retransmissions, as fewer ACKs would be sent back. Hence, the values for $N$ and $W$ must be suitably chosen.

C. Optimizing Parameters in PNC ARQ

Assuming homogeneous LSP (link successful probability) $p$ for both direct and overhearing links as in Section IV.B, we aim to find a proper set of $W$ and $N$ applicable for all $p$. In particular, let us require that the performance degradation from the benchmark throughput to be less than $e_0$ (e.g., 5%) for any $p$. This total throughput degradation $e_0$ comprises the feedback overhead degradation $e_1$, where $e_1 < e_0$ (e.g., $e_1 = 2.5\%$) and the wasteful retransmissions degradation $e_2$, where $e_2 < e_0$ (e.g., $e_2 = 2.5\%$), that is, $e_0 = e_1 + e_2$. Recall from the discussion in Section IV.C that when $p$ is smaller than 0.57, we would be better off using the traditional hop-by-hop transmission scheme. Hence, we only consider $p$ in the interval (0.57, 1).

We first derive the expression of $e_1$, the degradation due to feedback overhead $H$, in terms of $W$ and $N$. For each round, the probability for the destination to receive a coded packet (from the source through the relay) is $p^2$. On average, for each batch of $N$ successful receptions, the source node needs to transmit $N/p^2$ times. Thus, the average overhead due to ACK is $H = \frac{T_a}{(N/p^2)T_d}$, where $T_a$ is the transmission time of one ACK packet, and $T_d$ is the transmission time of one Data packet.

An ACK packet contains two parts: one part is the $W/8$ bytes used as the indicators in SACK; the other part is the packet header consisting of $K$ bytes. Suppose that a data packet has $D$ bytes, then we get

$$e_1 = H = \frac{K+W/8}{(N/p^2)\cdot D} = \frac{K}{(N/p^2)\cdot D} + \frac{W/8}{(N/p^2)\cdot D} \cdot (2)$$
In general, $e_2$, the degradation caused by wasteful retransmissions relative to total transmissions, also depends on $W$ and $N$. Unfortunately, a closed-form expression for $e_2$ as a function of $W$ and $N$ is difficult to derive and we have not succeeded in doing so. In particular, at the moment of an ACK wrap back (see definition in the last paragraph of Section VI.A) the packet in each window slot (except the left boundary slot, which is always an unACKed packet) is either a packet that has been received and acknowledged (referred to as an RA packet) or a packet that has not been received (referred to as an NR packet). The probability of boundary wrap back (see second paragraph of Section VI.A) in the next transmission cycle, which causes wasteful retransmissions, depends on $N$ and the number of NR packets (there can be anywhere between 1 and $W$ NR packets). After a boundary wrap back, besides RA and NR packets, another type of packets is possible: the packets that have been received but not yet acknowledged (referred to as an RNA packet). Since the source node cannot tell which packets are NR and which packets are RNA, it would transmit both types of packets. The transmissions of RNA packets are wasteful retransmissions. To count the wasteful retransmissions, we need to analyze the probabilities of $W^3$ states (there are $W$ slots in the window and each has three possible states, RA, NR, RNA). Although a Markov model can be drawn up for a specific pair of $W$ and $N$, the number of states in the Markov model will vary according to the $W$ and $N$ values. Thus, for each $(W, N)$, there is a Markov chain, the transitions of which are rather complicated. Overall, Markov model analysis is intractable.

Lacking a closed-form expression for $e_2$, here we provide a heuristic to find a proper set of $W$ and $N$ to approach the minimal throughput degradation. As $e_1$ can be found in closed form, we first limit the scope of $W$ and $N$ based on $e_1$. We need to determine the appropriate $W$ and $N$ that can make $H$ small. Note that (2) is an increasing function of $p$. To cover all possible $p$, we derive $W$ and $N$ based on the largest possible $p$ ($p=1$). Setting $p=1$ in (2) gives

$$H=\frac{K}{N \cdot D} + \frac{W/8}{N \cdot D} = \frac{1}{ND} \left( K + \frac{W}{8} \right)$$

(3).

Then we have

$$e_1 = \frac{1}{ND} \left( K + \frac{W}{8} \right)$$

(4).

The above shows the tradeoff between $W$ and $N$ given a target $e_1$. For our heuristic, let $e_1 = e_1^{(1)} + e_1^{(2)}$, where $e_1^{(1)} = \frac{K}{ND}$ and $e_1^{(2)} = \frac{W}{8ND}$ — i.e., we can divide the overall overhead budget into two parts, $e_1^{(1)}$ and $e_1^{(2)}$. Since $e_1^{(1)}$ depends only on $N$ (but not $W$), we can find the required $N$ to meet the overhead target $e_1^{(1)}$. For example, we could set $e_1^{(1)}=1\%$ and find the $N$ accordingly.

Once $N$ is fixed, $e_1^{(2)}$ then depends only on $W$. In particular, the smaller the $W$, the smaller the $e_1^{(2)}$. However, smaller $W$ would lead to larger $e_2$, the throughput degradation due to wasteful re-
transmissions. On the other hand, to keep the target \( e_1 \), from (4), \( W \) should be upper bounded by \( 8e_N D - 8K \).

Next, let us estimate intuitively how \( e_2 \) may depend on \( W \) (as mentioned earlier, the derivation of the exact closed-form expression is difficult). On average, the destination node sends back an ACK after every \( N/p^2 \) transmissions. And the probability for the source node to successfully receive an ACK is also \( p^2 \). Here we assume transmitting ACK packet in the reverse link has the same LSP. For example, both LSPs of links R-A and A-R are \( p \). On average \( 1/p^2 \) ACKs are needed for the source node to successfully receive an ACK. So on average totally \( N/p^4 \) data packet transmissions are needed before an ACK is received. This means \( W \) should be inversely proportional to \( p \). The smaller the probability \( p \) is, the larger the window size \( W \) needs to be. To cover all possible \( p \), we should derive \( W \) based on the smallest meaningful \( p \) (\( p=0.57 \), see first paragraph of Section VI.C).

In the following, given fixed \( N = \frac{K}{e_1} D \) and \( p=0.57 \), we derive the proper \( W \) through simulation. We first simulated a setting in which \( W \) is equal to the upper bound \( 8e_N D - 8K \) and get a throughput. After that, we gradually decrease \( W \) until the throughput begins to decrease. From massive simulations, we found that when \( W \) is not significantly smaller than \( 8e_N D - 8K \), the throughput degradation due to wasteful retransmissions is little and not sensitive to the change of \( W \). For this region, the throughput depends more on the overhead \( H \) and is an increasing function of \( W \). (i.e., second item in (2)) decreases. As \( W \) is decreased further, there comes a point where wasteful retransmissions become significant and that the maximum throughput is achieved at a \( W \) that strikes the right balance between overhead \( H \) and wastefulness due to retransmissions.\(^5\)

Our simulation results show that the above heuristic is an effective method to find a good set of \( W \) and \( N \). For example, when \( K=30 \) bytes (For the ACK frame in IEEE 802.11, \( K \) is around 30) and \( D=600 \) bytes (A conservative data packet size in wireless communication), \( N=4 \) \( W=170 \), the worst throughput degradation compared with the upper bound benchmark (calculated from throughput formula \( Th_1 \) in Section IV) is less than 4% for different \( p \). Further details on throughput comparison with the benchmark can be found in Table 2, where PNC ARQ with optimal set of \( W \) and \( N \) is referred to as PNC-O. Hence, with proper \( W \) and \( N \), we almost compensate for the loss due to the ACK lost and the ACK overhead.

On the other hand, for the simplistic system in which the transmission window \( W \) is 1, and in

\(^5\) Note that to find the proper set of \( W \) and \( N \), we can also first fix \( W \) and find the lower bound of \( N \) from (4); then directly increase \( N \) to reduce the overhead \( H \) until the proportion of wasteful retransmissions is large enough. However, intuitively, \( N \) is always much smaller than \( W \), and the step size of changing \( N \) may be too coarse to approach the optimal throughput.
which the receiver sends back an ACK upon each received data frame, the throughput degradation is very significant. In Table 3, we show the throughput gains of PNC ARQ with \( W=170 \) and \( N=4 \) relative to PNC ARQ with \( W=1 \) and \( N=1 \) under different \( p \). The throughput gains increase with the decrease of \( p \). When \( p=0.57 \), the gain is 167%.

| Table 2 | Throughput Comparison of Benchmark and PNC ARQ with Optimal Setting |
|---------|---------------------------------------------------------------|
| **Benchmark** (p/r) | **PNC-O** (p/r) | Degradation |
| \( p=0.95 \) | 1.73 | 1.67 | 3.5% |
| \( p=0.9 \) | 1.50 | 1.45 | 3.3% |
| \( p=0.85 \) | 1.31 | 1.27 | 3.1% |
| \( p=0.8 \) | 1.14 | 1.11 | 2.6% |
| \( p=0.75 \) | 0.99 | 0.97 | 2.0% |
| \( p=0.7 \) | 0.85 | 0.84 | 1.2% |
| \( p=0.65 \) | 0.73 | 0.72 | 1.4% |
| \( p=0.6 \) | 0.62 | 0.60 | 3.2% |
| \( p=0.57 \) | 0.57 | 0.56 | 2.1% |

| Table 3 | Throughputs of PNC ARQ under Different Sets of \( W \) and \( N \) |
|---------|---------------------------------------------------------------|
| \( W=1 \) \( N=1 \) (p/r) | \( W=170 \) \( N=4 \) (p/r) |
| **Gain** | 17% | 44% |
| \( p=0.95 \) | 15% | 48% |
| \( p=0.9 \) | 4% | 37% |
| \( p=0.85 \) | 0% | 27% |
| \( p=0.8 \) | 0% | 17% |
| \( p=0.75 \) | 0% | 12% |
| \( p=0.7 \) | 0% | 7% |
| \( p=0.65 \) | 0% | 1% |
| \( p=0.6 \) | 0% | 0% |
| \( p=0.57 \) | 0% | 0% |

| Table 4 | Throughputs under Different Link Successful Probabilities |
|---------|---------------------------------------------------------------|
| **PNC-NT** (p/r) | **PNC-O** (p/r) |
| **Gain** | 1.8% | 3.6% |
| \( p=0.95 \) | 0.5% | 6.7% |
| \( p=0.9 \) | 0.2% | 12.3% |
| \( p=0.85 \) | 0.1% | 17.9% |
| \( p=0.8 \) | 0.0% | 25.2% |
| \( p=0.75 \) | 0.0% | 33.4% |
| \( p=0.7 \) | 0.0% | 42.9% |
| \( p=0.65 \) | 0.0% | 56.0% |

Table 4 shows the throughput gains of PNC-O relative to PNC-NT. Recall that PNC-NT is the conventional PNC without stored-packet tracking. Here, for PNC-NT, we assume error-free ACK and neglect ACK air time. With this assumption, we are giving PNC-NT an advantage over PNC-O because we do not make the same assumption for PNC-O. We note that the throughput gains are nearly the same as those in Table 1 (Section IV), which compares PNC-T (with stored-packet tracking—as in PNC-O—but also assuming error-free ACK and neglecting ACK air time) with PNC-NT. In conclusion, with proper \( W \) and \( N \), the throughput gain from stored-packet tracking can be maintained in practical wireless systems with error-prone ACK and with ACK overhead.

### VII. General Applicability of ARQ Mechanisms

Recall from Section III that, for concreteness, this paper focuses on the cross atom in the ARQ design and analysis. The principles behind the ARQ mechanisms, however, are not limited to the cross atom only. In particular, they apply to all other atoms that exploit overhearing.

In this section, we discuss the general applicability of the ARQ mechanisms proposed in Sections IV to VI for other atoms. Our PNC ARQ contains three key ingredients in its design: 1) stored-packet tracking, 2) non-coupled ARQ, and 3) optimization of \( W \) and \( N \). We will discuss whether these three design ingredients can be applied to PNC atoms with overheard information.
(OPNC), PNC atoms without overheard information (Non-OPNC), SNC atoms without overheard information (OSNC), and SNC atoms without overheard information (Non-OSNC). When we say that a mechanism applies to a system, we mean the mechanism can increase the throughput of the system. Table 5 summarizes the general applicability of the three mechanisms for various set-ups.

|                  | OPNC | Non-OPNC | OSNC | Non-OSNC |
|------------------|------|----------|------|----------|
| Stored-Packet Tracking | Yes  | N/A      | Yes  | N/A      |
| Non-coupled       | Yes  | N/A      | Yes  | N/A      |
| W and N Optimization | Yes  | Yes      | Yes  | Yes      |

In the following, to illustrate the general applicability of the ARQ mechanisms to PNC atoms other than the cross atom of focus so far, we first consider the PNC atom as shown in Fig. 5, referred to as the star atom. Fig. 5 shows how the star atom consumes three timeslots to deliver one packet for each of three flows. Since the figure is self-explanatory, we will not go into the details here. The ARQ mechanism for the star atom will be explained in Part A.1 below.

![Fig. 5 PNC Star Atom](image)

**A. General Applicability of Stored-Packet Tracking**

As presented in Section IV.A, stored-packet tracking is a design that utilizes overheard information. Hence, stored-packet tracking is not applicable to atoms without overheard information (for both PNC and SNC atoms).

**A.1 General Applicability of Stored-Packet Tracking to OPNC**

Let us look at how stored-packet tracking can be applied to the star atom. With respect to Fig. 5, without loss of generality, we focus on destination D. Consider a particular round. Suppose that destination D only overhears $E_i \oplus F_i$ in the second time but fails to receive $A_i \oplus E_i \oplus F_i$ in the time slot in the 3-slot transmission pattern shown in Fig. 5. Proper tracking (see Section IV.A) requires destination D to check whether $A_i \oplus E_i \oplus F_i$ has been stored in its coded-packet pool (note:

6 For each PNC atom, there is a corresponding SNC atom. They have the same structure, but different transmission patterns: one utilizes PNC, one utilizes SNC. Generally, an SNC atom consumes more timeslots than its corresponding PNC atom.
could have been received in a previous round). By the same token, if destination D only receives \( A_i \oplus E_i \oplus F_i \) but fails to overhear \( E_i \oplus F_i \) in a round, for proper tracking it should check whether \( E_i \oplus F_i \) has been stored in its overheard-packet pool (see Section IV.A). As long as the unsuccessful packet (i.e., \( A_i \oplus E_i \oplus F_i \) or \( E_i \oplus F_i \) in the above) can be tracked and traced back to a previous successful reception of that packet, the desired packet \( A_i \) can be extracted. In general, all successfully received packets should be stored for potential use in the future. With our proposed stored-packet tracking mechanism (see Section IV.A), a systematic method is used to determine how long the packets should be stored so that they can be discarded when they are no more useful for extracting desired packets in the future.

For any PNC atom that exploits overheard information, to successfully extract its desired packet, the destination must XOR at least a coded packet (that embeds its desired packet) with an overheard packet (that can be used to extract the desired packet). Recall from the discussion in Section IV.A that, without tracking, the destination can extract its desired packet only when both the coded packet and the overheard packet are successfully received in the same round; with tracking, if the destination receives only a coded packet or an overheard packet that cannot be used to extract a desired packet by itself, the destination will store the coded packet or overheard packet for future use. In a future round, the destination may be able to receive an overheard packet or a coded that can be paired with the stored packet for the extraction of a desired packet. In other words, tracking allows overheard packets received from the past or the future to pair with coded packets, and this increases decoding opportunities at the destination node. In particular, this principle of tracking is common to all PNC atoms with overheard information, and therefore stored-packet tracking is applicable to all PNC atoms that exploit overheard information.

**Performance Gains of Stored-Packet Tracking on Various OPNC Atoms**

We now validate the throughput advantage of store-packet tracking in a number of OPNC atoms by simulations. Since the performance of a PNC atom also depends on whether coupled or non-coupled ARQ is adopted, we compare the PNC scheme with tracking (PNC-T) with the conventional PNC scheme without tracking (PNC-NT) under coupled ARQ (in Table 6) and non-coupled ARQ (in Table 7). We conducted simulations for all PNC atoms with overheard information, that is, atoms II, V (cross atom), VI, VII, VIII (star atom) and IX in [2]. Due to space limit, in this section we only present the simulation results under homogenous LSP (link successful probability). Also, here we only focus on the results under high LSPs, because we are interested in sce-

---

7 Since each PNC atom has different structure and transmission pattern (some of them are quite involved), complete presentation of all of them would occupy large space and be a diversion. We refer the reader to our paper [2] for details.
narios where PNC has better performance than traditional hop-by-hop transmissions (i.e., we want to compare different variations of PNC, but compare PNC versus the traditional scheme) — as discussed in the first paragraph of Section VI.C, PNC is more efficient than traditional hop-by-hop transmission only when LSP is above certain level (which may be different for different atoms). We also assume error-free ACK feedback and neglect the ACK transmission time in the simulations of parts A and B below.

**TABLE 6** SIMULATION RESULTS UNDER COUPLED ARQ

| LSP | Scheme | PNC-NT | PNC-T | Gain | PNC-NT | PNC-T | Gain | PNC-NT | PNC-T | Gain |
|-----|--------|--------|--------|------|--------|--------|------|--------|--------|------|
|     | P=0.9  |        |        |      | P=0.85 |        |      | P=0.8  |        |      |
| A   | II     | 0.67   | 0.70   | 5%   | 0.56   | 0.60   | 7%   | 0.46   | 0.51   | 11%  |
| T   | V      | 0.62   | 0.66   | 6%   | 0.50   | 0.56   | 12%  | 0.40   | 0.48   | 19%  |
| O   | VI     | 0.65   | 0.69   | 6%   | 0.48   | 0.54   | 12%  | 0.36   | 0.43   | 20%  |
| M   | VII    | 0.35   | 0.38   | 8%   | 0.25   | 0.30   | 20%  | 0.18   | 0.24   | 33%  |
|     | VIII   | 0.45   | 0.50   | 11%  | 0.32   | 0.39   | 22%  | 0.23   | 0.31   | 36%  |
|     | IX     | 0.33   | 0.49   | 48%  | 0.20   | 0.39   | 95%  | 0.12   | 0.31   | 158% |

**TABLE 7** SIMULATION RESULTS UNDER NON-COUPLED ARQ

| LSP | Scheme | PNC-NT | PNC-T | Gain | PNC-NT | PNC-T | Gain | PNC-NT | PNC-T | Gain |
|-----|--------|--------|--------|------|--------|--------|------|--------|--------|------|
|     | P=0.9  |        |        |      | P=0.85 |        |      | P=0.8  |        |      |
| A   | II     | 0.76   | 0.78   | 3%   | 0.66   | 0.69   | 5%   | 0.57   | 0.61   | 7%   |
| T   | V      | 0.73   | 0.75   | 3%   | 0.61   | 0.66   | 7%   | 0.51   | 0.57   | 11%  |
| O   | VI     | 0.86   | 0.90   | 4%   | 0.69   | 0.73   | 6%   | 0.54   | 0.60   | 11%  |
| M   | VII    | 0.47   | 0.50   | 6%   | 0.36   | 0.40   | 11%  | 0.27   | 0.33   | 22%  |
|     | VIII   | 0.61   | 0.65   | 7%   | 0.46   | 0.52   | 13%  | 0.35   | 0.43   | 23%  |
|     | IX     | 0.60   | 0.69   | 15%  | 0.43   | 0.55   | 28%  | 0.30   | 0.44   | 46%  |

The unit in Tables 6 and 7 is packets per time slot.

Tables 6 and 7 present the throughput gains of PNC-T over PNC-NT. As shown, the throughput gains due to stored-packet tracking for the OPNC atoms listed in the tables are all positive. Also, the throughput gain of an OPNC atom increases with the decrease of LSP. This is because tracking can increase decoding opportunities when wireless channel becomes worse (e.g., tracking will not offer any advantage when LSP=1).

A.2 General Applicability of Store-Packet Tracking to OSNC

The main difference between PNC and SNC is whether the network coding operation occurs at the physical layer or at a higher layer. Nevertheless, the same decoding operation (XOR) is performed in both for the extraction of the desired packet. For this reason, since stored-packet tracking can help to increase the chance of successful decoding where there is overhearing, it will also be useful in OSNC atoms that exploit overheard information. Thus, stored-packet tracking can also improve throughput in OSNC and is therefore applicable to OSNC.

B. General Applicability of Non-Coupled ARQ

In Section V, for the cross atom, we showed that non-coupled ARQ is more efficient than coupled ARQ with a Markov-model analysis. In principle, we can also construct such Markov
models for other atoms. As mentioned in Section IV.B, although the cross atom only has two traffic flows, the number of Markov states is already quite large, making the analysis quite complex. Most PNC atoms have more than two traffic flows, which mean they have even more Markov states, making the analysis even more complex. Therefore, here we compare the efficiency of non-coupled and coupled ARQs through massive simulation experiments.

**B.1 General Applicability of Non-Coupled ARQ to OPNC**

Let us take a look at how non-coupled ARQ can be applied to the star atom. Without loss of generality, we focus on flow A–D. When non-coupled ARQ is adopted, if source A successfully delivers a packet $A_i$ to the destination D in the current round, it will start to send the next packet $A_{i+1}$ in the following round regardless of the other flows. When coupled ARQ is adopted, if the source A successfully delivers a packet $A_i$ to the destination D but either flow B–E or flow C–F fails delivering its current packet $B_i$ or $C_i$ in the same round, it will still send the delivered packet $A_i$ in the following round. A source will not transmit the next packet indexed by $i+1$ until all sources have successfully delivered their packets indexed by $i$.

We have tested different atoms under different LSPs and the simulations are conducted without tracking (in Table 8) and with tracking (in Table 9). We found that for all PNC atoms with overheard information, non-coupled ARQ is more efficient than coupled ARQ whether packet tracking is adopted or not. As shown in Tables 8 and 9, the throughput gains range from 11% to 150%.

**B.2 General Applicability of Non-Coupled ARQ to OSNC**

In Section VII.A.2, we pointed out that an SNC atom does the same decoding operation as its corresponding PNC atom. In particular, whether an atom with a particular topological structure

\begin{table} 
\centering
\caption{Simulation results without stored-packet tracking}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
LSP & Scheme & $P=0.9$ & & & $P=0.85$ & & & $P=0.8$ & \\
\hline
& & Coupled & Non-Coupled & Gain & Coupled & Non-Coupled & Gain & Coupled & Non-Coupled & Gain \\
\hline
\multirow{4}{*}{A} & II & 0.67 & 0.76 & 13% & 0.56 & 0.66 & 18% & 0.46 & 0.57 & 24% \\
\hline
& V & 0.62 & 0.73 & 18% & 0.50 & 0.61 & 22% & 0.40 & 0.51 & 28% \\
\hline
& VI & 0.65 & 0.86 & 32% & 0.48 & 0.69 & 44% & 0.36 & 0.54 & 50% \\
\hline
\multirow{4}{*}{B} & VII & 0.35 & 0.47 & 34% & 0.25 & 0.36 & 44% & 0.18 & 0.27 & 51% \\
\hline
& VIII & 0.45 & 0.61 & 36% & 0.32 & 0.46 & 44% & 0.23 & 0.35 & 52% \\
\hline
& IX & 0.33 & 0.60 & 82% & 0.20 & 0.43 & 115% & 0.12 & 0.30 & 150% \\
\hline
\end{tabular}
\end{table} 

\begin{table} 
\centering
\caption{Simulation results with stored-packet tracking}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
LSP & Scheme & $P=0.9$ & & & $P=0.85$ & & & $P=0.8$ & \\
\hline
& & Coupled & Non-Coupled & Gain & Coupled & Non-Coupled & Gain & Coupled & Non-Coupled & Gain \\
\hline
\multirow{4}{*}{A} & II & 0.70 & 0.78 & 11% & 0.60 & 0.69 & 16% & 0.51 & 0.61 & 19% \\
\hline
& V & 0.66 & 0.75 & 13% & 0.56 & 0.66 & 17% & 0.48 & 0.57 & 19% \\
\hline
& VI & 0.69 & 0.90 & 29% & 0.54 & 0.73 & 32% & 0.43 & 0.60 & 37% \\
\hline
\multirow{4}{*}{B} & VII & 0.38 & 0.50 & 30% & 0.30 & 0.40 & 33% & 0.24 & 0.33 & 38% \\
\hline
& VIII & 0.50 & 0.65 & 30% & 0.39 & 0.52 & 35% & 0.31 & 0.43 & 40% \\
\hline
& IX & 0.49 & 0.69 & 40% & 0.39 & 0.55 & 41% & 0.31 & 0.44 & 42% \\
\hline
\end{tabular}
\end{table} 

The unit of throughput in Tables 8 and 9 is $p/t$: packets per time slot.
utilizes PNC or SNC, a specific destination (e.g., destination C in the cross atom) decodes the same coded packet (e.g., $A \oplus B$ in PNC or $A \oplus B$ SNC) from the relay with the same overheard packet (e.g., $B$ in PNC or $B$ in SNC). That means the packet delivery of a specific traffic flow (e.g., traffic flow A-C) depends on the overheard information from the same other traffic flow (e.g., traffic flow B-D). In other words, the dependencies among different traffic flows in an SNC atom are the same as in its corresponding PNC atom. For example, in both PNC and SNC, flow A-C is dependent on flow B-D in that to extract its desired packet $A$, destination C needs the overheard packet from source $B$.

Intuitively, since the dependencies among different traffic flows are the same, for any atom, as long as the atom utilizing PNC prefers non-coupled ARQ, the corresponding SNC atom would also prefer non-coupled ARQ. Furthermore, for a specific round, the possible throughput of adopting non-coupled ARQ relative to the possible throughput of adopting coupled ARQ are the same whether PNC or SNC is exploited. Hence, the throughput gains of non-coupled ARQ over coupled ARQ for PNC atoms (in Tables 8 and 9) are also applicable to SNC atoms.

### C. General Applicability of optimizing W and N

The performance of atoms other than the cross atom will also benefit from the optimization of transmission window size ($W$) and ACK frequency ($1/N$) in their ARQ. Take the PNC star atom for example. By the method proposed in Section VI.C, we found that a good set of $W$ and $N$ for the star atom are $W=130$ and $N=3$ when $K=30$ bytes and $D=600$ bytes (note: this set of ($W$, $N$) is different from the set for the cross atom). These $W$ and $N$ can ensure that the worst throughput degradation compared with the upper bound benchmark (i.e., the non-coupled simulation results in Table 9) is less than 4% for different $p$ (See Table 10). But if we just set $W=1$ and $N=1$, the throughput degradations range from 14% to 56% for the same $p$ range. Hence, proper optimizing $W$ and $N$ is important in the ARQ design of the PNC star atom.

### Generality of Optimizing $W$ and $N$

In general, optimizing transmission window size and ACK frequency is applicable to all network coded systems that make use of ARQ to ensure reliability; i.e., it is not just limited to our study of PNC here. By optimizing the transmission window size $W$ and the ACK frequency $1/N$, the performance of these systems can be significantly improved.

| Benchmark ($p/t$) | $p=0.95$ | $p=0.9$ | $p=0.85$ | $p=0.8$ | $p=0.75$ |
|-------------------|-----------|-----------|-----------|-----------|-----------|
| $PNC-O$ ($N=3$ $W=130$) ($p/t$) | 0.77 | 0.63 | 0.51 | 0.42 | 0.33 |
| Degradation1 | 3.4% | 2.9% | 2.1% | 2.4% | 2.6% |
| $PNC$ ($N=1$ $W=1$) ($p/t$) | 0.77 | 0.63 | 0.51 | 0.42 | 0.33 |
| Degradation2 | 14% | 23% | 33% | 42% | 56% |

$p/t$: packets per time slot.
we can reduce the throughput degradation due to lost ACK and ACK overhead. In a practical network-coded ARQ system, lost ACK and ACK overhead are part of the system, whether we are talking about a system operated with PNC or SNC. Thus, optimizing $W$ and $N$ can potentially minimize this throughput degradation in all network coded systems in which ACK is error-prone and ACK overhead cannot be ignored.

To our best our knowledge, there has been no past network-coded work that considered the issue of optimizing transmission window size and ACK frequency in a rigorous manner (e.g., [4] and [5]). Hence, going forward, it will be interesting to investigate how to set $W$ and $N$ in different network coded systems taking into consideration the effect of error-prone ACK and ACK overhead.

VIII. CONCLUSION

This paper has investigated ARQ (Automatic Repeat request) designs for PNC (Physical-layer Network Coding) systems. Specially, we focus on PNC systems that exploit overheard information.

Our investigation indicates that for PNC ARQ: 1) Proper tracking of stored coded and overheard packets can increase the chance of packet extraction, hence throughput. 2) Between two generic variations of PNC ARQ, coupled ARQ and non-coupled ARQ, non-coupled ARQ is more efficient than coupled ARQ. 3) Proper optimization of the transmission window size and ACK frequency can minimize wasteful retransmissions due to lost ACKs and the ACK feedback overhead, allowing throughput to approach an ideal upper bound.

Our throughput analyses and performance evaluations indicate that for our investigated atoms, stored-packet tracking combined with non-coupled ARQ yield considerable throughput gains. Moreover, by properly optimizing the window size and ACK frequency, we can nearly compensate for all the throughput loss due to lost ACK and ACK overhead (the resulting throughput degradation is within 4% of a theoretical throughput upper bound).\footnote{The three ARQ principles, stored-packet tracking, non-coupled ARQ, and window-size and ACK frequency optimization, are also applicable to SNC. In other words, they can also improve performance in SNC systems.}

As a first investigation of ARQ for PNC building blocks that exploit overheard information, this paper has only focused on end-to-end ARQ. Going forward, it will also be interesting to investigate link-by-link ARQ and compare the relative merits of these two ARQ approaches for PNC systems.

REFERENCES

[1] S. Zhang, S. Liew, and P. Lam, “Physical layer network coding,” ACM Mobicon’06, 2006
[2] J. He, S. C. Liew, “Building Blocks of Physical-layer Network Coding,” accepted by IEEE Trans. Wireless Communication.
[3] J. He, S. C. Liew, “ARQ for Physical-layer Network Coding,” Technical Report
[4] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, “XORs in the air: practical wireless network coding,” IEEE/ACM Trans. Netw., vol. 16, no. 3, pp. 497–510, 2008
[5] Georgios S. Paschos, Constantinos Fragiadakis, Leonidas Georgiadis and Leandros Tassiulas, “Wireless network coding with partial overhearing information,” in IEEE INFOCOM, 2013.
[6] Okada, H., Hiramuki, H., Mase, K., An ARQ Scheme Using NACK for Network Coding in Wireless Multihop Networks. 10th International Symposium, On ISADS 2011, March 2011.
[7] J. K. Sundararajan, D. Shah, and M. Médard, “ARQ for network coding,” in Proc. of IEEE International Symposium on Info. Theory (ISIT), 2008.
[8] D. Nguyen, T. Nguyen, and B. Bose, “Wireless broadcasting using network coding,” in Proc. NetCod, Jan. 2007.
[9] Q.-T. Vien, L.-N. Tran, and H. X. Nguyen, “Network coding-based ARQ retransmission strategies for two-way wireless relay networks,” in Proc. 2010 Intl. Conf. Software Telecommun. Comput. Network.
[10] L. Lu, L. You, Q. Yang, T. Wang, M. Zhang, S. Zhang, S.C. Liew, “Real-time implementation of physical-layer network coding”, SRIF ’13, 2013
[11] S. Wang, Q. Song, X. Wang, and A. Jamalipour, “Distributed MAC protocol supporting physical-layer network coding,” IEEE Trans. Mobile Comput., vol. 12, no. 5, pp. 1023–1036, May 2013.
[12] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, “Selective acknowledgment options,” RFC-2018, 1996.