

Security Vulnerabilities for Network Coding in Wireless Networks

Ruben Sethi
rsethi@cs.purdue.edu

I. INTRODUCTION

Network coding is a relatively new area of research which has been shown to have a large number of benefits over the routing and forwarding methods which are most commonly used today. Current work in network coding is most generally aimed toward achieving a higher effective throughput in wireless networks. In this paper, we will demonstrate some of the very simple security vulnerabilities which exist in the current network coding schemes. However, before getting into the details of all the vulnerabilities, the basic concept of network coding must be well understood.

In traditional networks, every intermediate node in a source to destination path simply stores and forwards packets to the next hop, unchanged. Network coding is a system in which nodes in a path will mix packets before forwarding them out. The mixing of packets is achieved by either making random linear combinations of packets, or XORing packets. After the combining of packets takes place, these packets will have to be extracted from the combined, or *coded*, packets.

Current network coding systems are very well optimized for performance, but these systems are not well suited for practical environments. These protocols achieve very good performance gains over traditional routing, but due to the use of network coding, these protocols have a new set of security vulnerabilities which need to be dealt with before these systems can be practical. This paper takes a closer look into some of the vulnerabilities that have not yet been formally analyzed and shows just how serious these security threats can be.

The rest of the paper is divided into 5 sections. Section 2 will give a more in-depth explanation of network coding and the protocols used for our simulations. We then go into the basic description of the adversarial

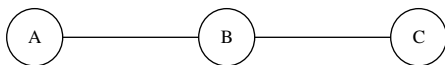


Fig. 1. Simple Topology

model for our research in Section 3, followed by a summary of the attacks which are analyzed by this paper in Section 4. We end the paper with the results of our experiments (Section 5) and the conclusions supported by these results (Section 5). Preliminary results of our work was recently published in the Workshop of Network Security Protocols in ICNP 2008 [1].

II. A CLOSER LOOK AT NETWORK CODING

The quickest and most effective way to understand the benefits of network coding is through a concrete example. This section will start off with a very simple example of network coding, demonstrating the significant benefits over traditional routing. We will continue with an explanation of the protocols used for testing the attacks against network coding schemes.

A. Network Coding Basics

Fragouli et al. [2] explain a good example which we will also walk through for the first look into network coding. Consider a simple wireless network with 3 nodes, as shown in Figure 1. Given 2 flows, flow f_1 with source A and destination C, and f_2 with source C and destination A. In traditional routing, the sequence of transmissions shown in Figure 2 would take place, where $X \in f_1$ and $Y \in f_2$. This results in a total of 4 total transmissions for one packet to reach each of their respective destinations.

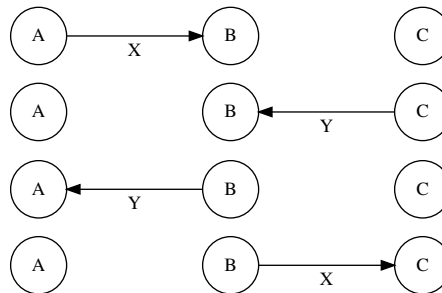


Fig. 2. Traditional Scenario

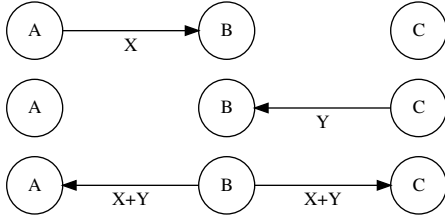


Fig. 3. Network Coding Scenario

Now consider the case where network coding takes place. In this example, node B will XOR the packets X and Y , then broadcast this coded packet to both of the other nodes. Nodes A and C store the packets which they have transmitted, X and Y , and so both of these nodes have everything they need in order to *decode*, or extract the unknown packet from, the coded packet. Since both nodes A and C have one of the packets contained in the coded packet, each node can extract the unknown packet by simply doing an XOR operation between the coded packet and the packet which is known. For example, node A will receive the coded packet $X \oplus Y$, and will have packet X stored. In order to get packet Y from the coded packet, node A will XOR the coded packet $X \oplus Y$ with the known packet X , giving the equation:

$$(X \oplus Y) \oplus X = Y$$

This results in the sequence of transmissions shown in Figure 3, which has a total of 3 transmissions. In this scenario, there is a 25% reduction in the number of transmissions, which would obviously increase the overall throughput of the two flows.

This is an very simplified example of when network coding would help, but it is a good basis for beginning to understand the advantages network coding.

Network coding schemes vary widely in the method used to route and find chances to intelligently mix packets, otherwise known as *coding opportunities*. Keeping in mind the wide variety of schemes, there are two major classes of protocols. These distinct approaches are called *Intra-flow Network Coding* and *Inter-flow Network Coding*.

B. Intra-flow Network Coding

The protocols which fall into the intra-flow class of network coding all have one thing in common: the packets which are coded together in these schemes are only from the same flow. Given this condition, the simple example covered in the previous section is not one which demonstrates this idea. We can demonstrate this idea by taking a look at one intra-flow network coding protocol which we used for testing some of our attacks.

The intra-flow network coding routing protocol we used was MORE [3]. This protocol defines methods for both routing and forwarding packets for each flow. For a given flow, MORE defines an algorithm for intelligently choosing nodes in the wireless network which have a lower ETX [4] metric. These nodes are chosen in order to maximize opportunistic overhearing in the network and minimize the interference among nodes in order to achieve a higher throughput.

Given a flow f of n packets such that $f = \{p_1 \dots p_n\}$, the flow f is broken up into batches of k packets by the source. With each batch of k packets, we say batch 1 of a flow f consists of the packets p_1 to p_k , batch 2 would then consist of packets p_{k+1} to p_{2k} , and batch x contains packets $p_{(x-1)k+1}$ to p_{xk} . Once the flow is broken up into batches, the source will start broadcasting coded packets from the first batch of packets. Each coded packet C will be a random linear combination of the packets within the batch, such that

$$C = \sum_{i=1}^k c_i p_i$$

where each c_i is a random coefficient. The source will keep broadcasting these coded packets until it receives an ACK from the destination. This ACK will signal the source to start broadcasting packets from the next batch, and the process will start all over again with this new batch of packets.

Upon receiving these coded packets from either the source or other forwarding nodes, a receiving node which is in the forwarding set of nodes for this flow will check whether or not the packet is *innovative*. If a packet is innovative, it contains new information about the packets in a flow, meaning that this linear combination of *native*, or uncoded, packets cannot be derived from any of the previously received packets. If a forwarding node receives an innovative packet, it will store it. Otherwise, the received packet is dropped. A forwarding node will then make random linear combinations of the stored coded packets and re-broadcast these new coded packets to the downstream nodes. These linear combinations of coded packets are still actually simple linear combinations of native packets, due to the distributive property of multiplication.

When the destination receives each coded packet, it will check whether or not the packet is innovative yet again, dropping all non-innovative packets. Once the destination has received a sufficient number of innovative packets, it will be able to decode all of the native packets from these coded packets by solving a system of equations. Once it has the ability to decode all the native packets in the current batch, the destination will unicast an ACK to the source. This whole process will repeat until the destination has successfully received and

decoded all of the packets in the flow.

C. Inter-flow Network Coding

Intuitively, inter-flow network coding protocols are the ones in which packets in separate flows are coded together. The scenario described in the previous section is a valid example of this class of protocols. For our experiments with these protocols, we chose to use DCAR [5], which is based on the COPE [6] system.

In the COPE protocol, no new method of routing is specified. Under this scheme, a node c will simply code together packets from flows $f_1 \dots f_n$ if the next hops for these flows, from c , have a high probability of overhearing the packets from all of the $n - 1$ other flows. In other words, the packets are coded together if the next hops have a high probability of being able to decode these coded packets into the native packets.

In [5], the authors point out that COPE allows for only very restricted coding opportunities. Packets are only coded for 2 hops at a time, and there is no optimization for attempting to maximize the number of coding opportunities. DCAR improves on both of these major downfalls. The authors of [5] come up with a Coding-aware Routing metric, or CRM, which aids in routing flows to maximize coding opportunities. This paper also removes the restriction of a 2 hop coding region by allowing coding at a node as long as there is a high probability that *any* node which is downstream from the coding node has a high probability of being able to decode the packets. Our experiments use ETX [4] routing instead of the CRM routing scheme described in [5]. As a result, the number of coding opportunities decreases. However, Section 5 will show that the effects of the attacks can only become more detrimental with an increasing number of coding opportunities.

III. ATTACK MODELS

For each class of network coding protocols, we tested the effects of separate attacks. As a result, we have two separate adversarial models; one for each class of network coding protocols. Each one of these models makes very few assumptions.

A. Intra-flow Attack Model

For our intra-flow network coding attack model, we assume the following:

- All nodes in the network are static nodes; there are no mobile nodes.
- For a given flow, only the forwarding nodes, and neither the source or destination, can be attackers.
- Once a path between source and destination is determined, there is no mechanism to improve the throughput of the flow by dynamically changing the path.

- There is no authentication, or any other security mechanisms, for the link-state routing messages used to determine ETX weights.

B. Inter-flow Attack Model

The assumptions for the inter-flow attack model are very similar:

- The wireless network is static.
- For a given flow, the source and destination nodes cannot be attackers.
- There are no authentication mechanisms for ensuring a packet is from the node which the packet is labeled to be from.
- There is no mechanism for determining whether or not a packet is polluted or not.

Given these constraints, the results of our experiments will show the devastating effect of some very simple attacks.

IV. ATTACKS

As stated earlier, this paper analyzes two different classes of attacks for the two types of network coding protocols.

A. Intra-flow Attacks

For intra-flow network coding, we analyzed five attacks. Each one of these attacks negatively affects flows through dropping packets. The most basic attacks we will look at are the *data-dropping*, *ACK-dropping*, and *data-and-ACK-dropping* attacks.

A data-dropping attacker node simply drops every data packet it receives, and an ACK-dropping attacker does the same, except it only drops ACK packets. Intuitively, a data-and-ACK-dropping attacker drops every packet it receives for a flow.

The other two attacks analyzed in this paper are modifications of these three basic attacks. These attacks are *Local Metric Manipulation* and *Global Metric Manipulation*.

Local Metric Manipulation is when an attacker advertises that it has 100% link quality with all of its one-hop neighbors, while Global Metric Manipulation is when an attacker node advertises 100% link quality with the destination of a flow. Both of these attacks manipulate the link-state metrics in the network, as well as drop all data and ACK packets in a flow.

B. Inter-flow Attacks

In the inter-flow network coding scenario, we test two general *packet pollution* attacks. A packet pollution consists of taking a received packet from a flow, or creating a packet which is disguised to be part of some flow, and changing the payload to a garbage value in an attempt to disrupt the communications between a source

and destination. Our simulations only have one type of pollution attack. The attack is very simple; If an attacker is part of a path for a given flow, the attacker will pollute the packets of that flow with a specified frequency.

The effects of the pollution attack against these flows should be closely observed. Since inter-flow network coding combines packets across flows, the pollution of one packet can lead to the pollution of more than one flow. The greater the number of flows being coded together, the greater the chances that pollution of one flow can contaminate another flow. Therefore, the decreases in throughput shown in the results of our experiments for inter-flow pollution attacks are likely to be amplified by the use of any mechanism to increase the coding opportunities, such as the CRM in [5].

V. EXPERIMENT RESULTS

The inter-flow network coding attacks were implemented using the GloMoSim [7] wireless network simulator and the Roofnet [8] topology for our simulated network. This consists of 38 nodes using 802.11a/b MAC with 5.5 Mbps raw data transmission rate. For each simulation, we chose 20 flows with randomly picked source-destination pairs. Each source sent out packets at a rate of 128 kbps. As stated earlier, these experiments were run using a simplified version of the DCAR [5] protocol. The results for the inter-flow network coding attacks have not yet been completed, so we cannot present or analyze any data for the corresponding attacks. The inter-flow network coding attacks will be omitted from this section onward.

For our intra-flow network coding experiments, we used an implementation of MORE[3] using the GloMoSim wireless network simulator and the Roofnet [8] topology also. For each simulation, each of the source-destination pairs were chosen randomly. These different types of attacker, as defined earlier, are denoted as *Drop-Data*, *Drop-ACK*, *Drop-All*, *LMM*, and *GMM*, respectively. All flows were unicast, and all attacker nodes were chosen randomly among the nodes that are in the forwarding set of each flow, with the exceptions

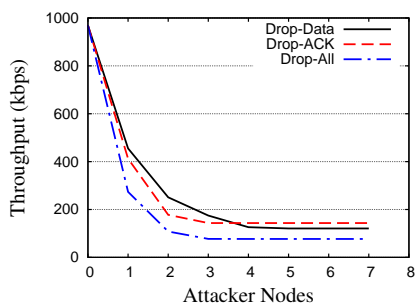


Fig. 4. Average Throughput with Attacker Nodes

of the LMM and GMM attacks, in which the attackers were chosen from the set of all 38 nodes.

We analyze the effects of each type of attack separately. For the packet dropping attacks, we have an in-depth analysis of the cases with one attacker node. Figure 4 shows the average throughput of flows with the number of packet dropping attackers ranging from 0 to 7, while Figures 5 and 6 show comparisons between flows with no attackers and those with only one packet dropping attacker. The effects of the LMM attack were not notably significant. For this reason, we did not include an analysis of the attack. The results of the GMM attacks are shown in Figure 7, with the number of attackers varying from 0 to 7.

A. Data Dropping

As shown in Figure 4, only dropping data packets can have a very drastic effect on the average throughput. Having multiple attacker nodes can result in dropping the overall average throughput to about 12% of normal throughput. However, the most significant throughput decrease is between the simulations run without attackers and those with only one attacker. Having only one attacker dropped the average throughput down to roughly 47% of the original throughput.

From just the averages, we can tell that packet dropping attacks can be detrimental to the MORE protocol, but what is the distribution of these results? Figure 5 contains the answer. Over 50% of the flows resulted in a throughput of less than 500 kbps, despite the lowest throughput for any flow not under attack being around 524 kbps, and the median throughput for having no attack being about 970 kbps.

From Figure 6(a) it is clear that very few flows were left completely unaffected from this attack. Although many flows were minimally affected, there were also many cases in which the throughput dropped to zero. This is surprising, considering there is only one attacker in these cases. We found that even though there were many different paths between the source and destination nodes in these flows, there were nodes through which all packets had to pass. As a result, the throughput

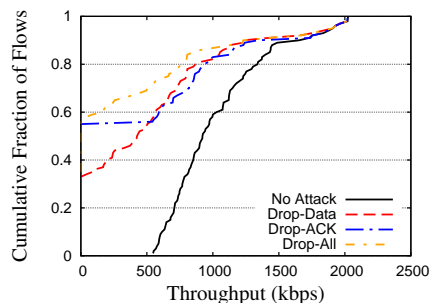
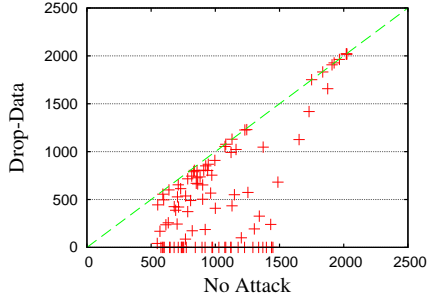
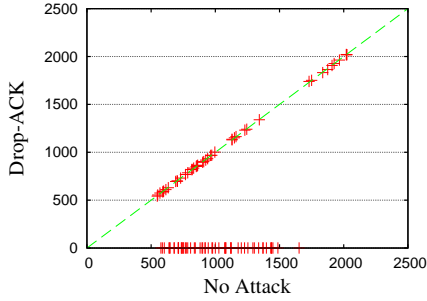


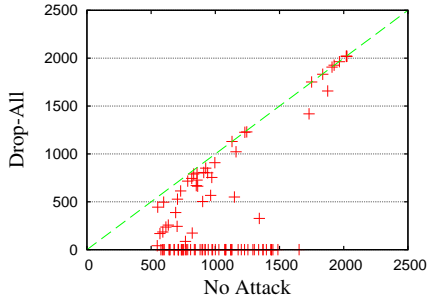
Fig. 5. Throughput CDF with One Attacker



(a) Data Dropping



(b) ACK Dropping



(c) Data and ACK Dropping

Fig. 6. Throughput Scatter Plots for One Attacker Node

dropped to zero when these nodes were data dropping attackers. In general, the greater the number of unique paths from source to destination, the smaller the effect of the attacker nodes. The only exception to this is the flows that were unaffected. These flows usually had the source and destination within range of each other, which means they had no forwarding set, and, in turn, had no nodes to choose as attackers.

B. ACK Dropping

Flows with only ACK dropping attackers had rather interesting outcomes. As shown in Figure 4, the effects of the ACK dropping attackers were very similar to those of the data dropping attackers. The average throughput for the flows with these attackers leveled off slightly higher than that of the flows with data dropping attackers, but when comparing the cases with only one attacker, ACK dropping attackers seemed to have the upper hand.

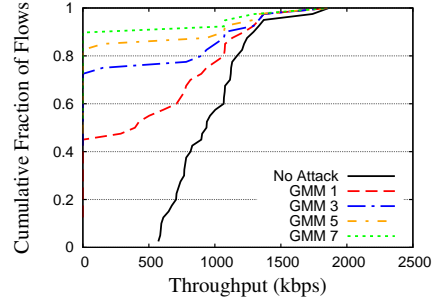


Fig. 7. Throughput CDF with GMM Attackers

Having only one ACK dropping attacker in the flows, the average throughput dropped to about 42% of the original throughput.

Over 50% of the flows with ACK dropping attackers resulted in a throughput of practically zero. The rest of the flows were left unchanged by the ACK dropping attack, as shown in Figures 5 and 6(b). This is because the ACK dropping attack only has an effect if the attacker node is on the ACK path. If the ACK dropping node is on the ACK path, the destination node will send the ack, but the source node will never receive it. In turn, the source will only send packets from the very first batch in the flow.

C. Data and ACK Dropping

After reviewing the results of the individual data and ACK dropping attacks, the results from the integration of these two attacks is rather intuitive. As we can see from Figure 4, the flows with this attack had the lowest throughput, on average. With only one attacker dropping both ACK and data packets, the average throughput is dropped to 28% of that with no attack.

As Figure 5 shows, dropping both data and ACK packets resulted in around 57% of all flows to have a practical throughput of zero, while 70% of all flows under this attack resulted in a throughput of about 500 kbps or less.

Figure 6(c) illustrates the outcomes of each of these flows having one attacker dropping all packets. As we can see, the graph seems to be an integration of Figures 6(a) and 6(b). Any flow that was affected by only the ACK dropping attack was also affected in the same way for this attack. The rest of the flows were affected the same as they were for the data dropping attack.

D. Global Metric Manipulation

The Global Metric Manipulation attack is when the attacker nodes create a false link between themselves and the destination nodes. This attack could be done by strategically placing an attacker node adjacent to the destination, and that adjacent attacker would make the destination seem to have a 100% link quality with the

other, randomly chosen, attacker. The randomly chosen attacker would drop both ACK and data packets for maximum effect. Essentially, this attack increases the likelihood of the randomly chosen attacker to be in the forwarding set or ACK path, making the packet dropping attacks more likely to be effective, since the packet dropping attacks need to be in either the forwarding set or the ACK path in order to have any effect.

As shown in Figure 7, this attack was extremely effective. This figure shows the throughput CDF of these attacks for one attacker, *GMM 1*, to seven attackers, *GMM 7*, compared to the throughput with no attack. Intuitively, this attack is going to be more effective than any packet dropping attack alone. With only one attacker, over 40% of the flows have an effective throughput of 0 kbps. This attack gets progressively more effective as the number of attackers increases, dropping around 90% of flows to 0 kbps throughput.

Since the attacker nodes for these simulations were chosen using a method different than that of the other simulations, the results observed from these attacks cannot be directly compared to those observed from the packet dropping attacks.

VI. CONCLUSION

Early speculations on network coding supported the idea that network coding is naturally resilient to packet dropping due to the redundancy provided by the random linear combinations of packets during their traversal of the intermediate nodes from source to destination. As we have just shown, this is not the case. Even a single packet dropping attacker can be detrimental to the throughput of a flow using the MORE protocol. Our work shows that this is a serious threat that needs to be considered. It can take only one attacker node to drop the average overall throughput down to around 30% of normal throughput.

REFERENCES

- [1] J. Dong, R. Curtmola, R. Sethi, and C. Nita-Rotaru, "Toward secure network coding in wireless networks: Threats and challenges," *Secure Network Protocols, 2008. NPSec 2008. 3rd IEEE Workshop on*, Oct. 2008.
- [2] C. Fragouli, J. L. Boudec, and J. Widmer, "Network coding: An instant primer," in *ACM SIGCOMM Computer Communication Review, Vol. 36*, 2006.
- [3] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. of ACM SIGCOMM '07*, 2007.
- [4] D. S. J. D. Couto, J. B. D. Aguayo, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proc. of MOBICOMM '03*, 2003.
- [5] J. Le, J. C. S. Lui, and D. M. Chiu, "DCAR: Distributed coding-aware routing in wireless networks," in *Proc. of ICDCS '08*, 2008.
- [6] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *Proc. of ACM SIGCOMM '06*, 2006.
- [7] "Global mobile information systems simulation library - glomosim," <http://pcl.cs.ucla.edu/projects/gloimosim/>.
- [8] "MIT roofnet - publications and trace data." <http://pdos.csail.mit.edu/roofnet/doku.php?id=publications>.