Neighborhood Monitoring in Ad Hoc Networks

Axel Krings
University of Idaho
Moscow, ID 83844-1010
krings@uidaho.edu

Stephan Muehlbacher-Karrer
University of Idaho
Moscow, ID 83844-1010
stephanm@cs.uidaho.edu

ABSTRACT

Much research in ad hoc networks utilizes monitoring mechanisms to detect maliciously acting nodes. The faults that a system is exposed to range from simple omissions to packet manipulations and misrouting. Several monitoring techniques have been presented that address malicious behavior in specific environments, but each has their restrictive assumptions about detection and mitigation in hostile environments. We generalize the notion of malicious behavior of related research and outline a k-hop monitoring approach that allows us to drop many assumptions limiting previous work. Specifically, we include collaboration of malicious nodes, the presence of malicious nodes during neighborhood exploration, and misrouting. However, the approach exposes also the associated overhead in providing detection and correction thresholds.

1. INTRODUCTION

Wireless technology has found its way into many applications such as LAN, wireless control, SCADA, fast deployable emergency communications systems, UAV communication, not to mention a wide range of consumer products. Many protocols are used, e.g., bluetooth, zigbee or 802.11.

Wireless technology is a mixed blessing. On one side it is very convenient and flexible, as its communication media is the ether. On the other side there are many concerns related to reliability, security and survivability. Due to the wireless broadcast, packets can be received by any node that can receive the signal. Furthermore, nodes may move out of range, get captured, unwanted nodes may try to join or signals may be jammed. The attack space is large and difficult or impossible to enumerate. However, one can say that, in general, all attacks in the end amount to manipulation of packets, such as delaying, dropping, modifying, fabricating, misrouting, or sniffing of packets.

The goal is of course to resolve issues related to the above manipulations and the first step is to understand the impact of faults, i.e., a common model is needed that allows for the identification of what faults can occur and be tolerated. The key issues are the fault model and fault detection/masking mechanisms.

1.1 Background

A wireless network can be represented as a digraph $G = (V, E)$, where V is a finite set of all wireless communication nodes n_i and E is the set of edges e_{ij} representing communication capability between nodes n_i and n_j. The left part of Figure 1 [6] depicts a wireless network consisting of four nodes, and for each node the range of the broadcast is indicated by the associated oval. The network graph for the four nodes is shown on the right side. Node n_1 can receive the signal from n_2 and vice versa. Thus e_{12} and e_{21} are in E. Node n_3 can only receive from n_1 but its signal is not strong enough to reach it, indicated by e_{13}. Node n_4 can neither receive nor be received by any other node. Note that this graph presentation captures the nature of any antenna behavior, e.g., omni directional or directed.

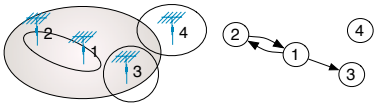

Figure 1: Network graph

Since our primary concern is packet manipulation within the network, we are interested in any mechanism that allows us to identify such manipulations in the hope to detect and mitigate. The key mechanism for detecting faults is redundancy, which can be *information*, *time* or *spacial* redundancy. For example, cyclic redundancy checks can identify bit errors. Timeout mechanisms can initiate resending of packets and "redundant packets" can be used to detect manipulation. It is the latter that is of special interest, as packet redundancy in wireless networks has no extra cost and it can be effectively exploited by monitoring. Specifically, any node in range can see a packet. When a node n_m monitors another n_i it suffices for n_m to buffer a packet sent to n_i to determine if n_i forwards it modified, or forwards it at all. Such simple cross-monitoring in wireless networks has been addressed in the context of watchdogs [7], a mechanism where one node is a watchdog for certain behavior of another node, e.g., to see if a node actually forwarded a packet it had received.

The first issue in monitoring is to determine the scope of monitoring. We will introduce k-path monitoring, in which a monitor monitors the interaction of nodes over k hops. The second issue is the subgraphs that neighborhoods span, e.g., cliques versus partially connected subgraphs.

1.2 Related work

The first question about faults in wireless network is: which faults are to be addressed? This brings up fault models. One description of fault model is that of [1] where benign faults and transmissive & omissive versions of symmetric and asymmetric faults are discussed. Whereas a benign fault is globally verifiable, a symmetric fault is a value fault, and an asymmetric fault is a Byzantine fault. The notions of transmissive and omissive for the last two fault types addressed whether the fault behavior was due to an actual manipulation or an omission.

The simplest case of monitoring is the watchdog mechanism described in [7], where "misbehaving" nodes causing omission faults were detected by so-called "watchdogs". The concept was extended in [8] where collaborating groups of malicious nodes were considered. In [2] the effectiveness of various watchdog schemes was investigated. Their results suggest that watchdog schemes are indeed able to detect a number of attacks such as omissions and certain symmetric faults but expose limitations, e.g., fabrication of false route error messages. Wormhole attacks were addressed in [9], where statistical analysis was used for detection of nodes which launch them. Detection of malicious behavior due to observation of monitoring nodes operating in promiscuous mode was shown in [3]. Watchdogs are not limited only to forward monitoring. For example in [4] an extended watchdog mechanism was presented that implemented backward monitoring based on CTS and RTS messages at the MAC layer.

The concept of monitoring can be extended to neighborhood monitoring, in which groups of monitors form a neighborhood. In [5] and [6] the impact of neighborhood monitoring was exploited. In [5] authenticated neighborhoods were considered. However, the work assumes an attack-free environment during neighbor discovery, i.e., no malicious nodes exist before the completion of the neighbor discovery. Furthermore, no misrouting attacks are considered. It does not address network dynamics nor collaboration between malicious nodes, i.e., the probability of framing due to collaborating nodes is assumed small and based on statistical arguments. We diverge from this assumption and consider pathological behavior as likely. The impact of neighborhood watching was considered in [6] to analyze the impact of topology on reliability. However, this was based only on the general structure of the neighborhood, i.e., the width of adjacent neighborhoods represented as a join-graph, in order to determine the reliability and survivability of a link.

1.3 Contributions

This work has several contributions: (1) It extends the notion of neighborhood monitoring of [5, 6] to general k-path monitoring (for $k = 1, 2$), which makes no restrictions on the capability of malicious nodes. (2) It introduces a monitoring approach able to capture the assumptions of previous work in one model, thereby allowing comparison between approaches. (3) It addresses fault detection and recovery for specific identified fault types. (4) It outlines the protocol that achieves the above.

2. MULTI-HOP NEIGHBORHOOD WATCH

In this section we define a graph-based neighborhood watch mechanism which extends the neighborhood described in [5] as will be identified. A neighborhood is a collection of nodes involved in monitoring their neighbors. In order to avoid malicious nodes from uncontrolled involvement in neighborhoods, one could assume that neighborhoods are authenticated, e.g., by using the neighborhood authentication methods used in [5]. However, contrary to [5] we assume that malicious nodes may be admitted into a neighborhood. Authentication would simply disallow "just any node to attack out of the dark".

2.1 Attack Model

An attack may originate within a node that is part of the authenticated neighborhood or not. We consider the following attacks:

1. Attack from outside of the authenticated neighborhood, i.e., non-authenticated nodes. Any involvement from such nodes will be discarded and carries no weight.
2. Attack from a good node gone bad within an authenticated neighborhood. Here the node was authenticated as part of a neighborhood discovery.
3. Attack from a malicious node that joint the neighborhood.

Malicious nodes may exhibit two kinds of behavior, i.e., *passive* and *active*. A passive mode does not cause packets to be manipulated, but it accuses other nodes, thereby trying to frame them. An active node will cause the following problems: omission (o), delay (d), routing (r), fabrication (f), and content manipulation (m). Denial of service or jamming can be addressed by naturally expanding the concept here to disjoint paths. Thus the set of fault types $\mathcal{F} = \{o, d, r, f, m\}$.

2.2 Evolving Neighborhoods

Figure 2 shows the evolution of primitives used in neighbor monitoring schemes in four graphs. A simple watchdog[1], e.g., [7], is shown in Figure 2a), where node n_r can monitor n_s to detect $\mathcal{F} = \{o, d, m\}$. This allows for backward recovery but not forward recovery, i.e., backward notification allows n_r to resend or send the message via another path. Also, to satisfy $\mathcal{F} = \{d, m\}$ the packet under observation needs to be buffered at n_r sufficiently long. For delayed packets this of course puts restrictions on the buffer size. The backward monitoring of [4] uses the primitive in Figure 2b), where n_t can detect $\{o, d\}$, of n_r's packet at n_s, and n_r can detect $\{o, d, m\}$. The neighborhood monitoring scheme of [5] uses primitive Figure 2c). Under their "fault-free neighborhood discovery" assumption monitors n_r and n_m allow detection of $\{o, d, f, m\}$. Node n_m is drawn thick to indicate that it can be expanded to represent multiple monitors. The monitoring primitive used in this research is k-hop monitoring. For $k = 1$ and $k = 2$ the primitives are depicted in Figure 2c) and d) respectively. The fault sets of this approach will be discussed below.

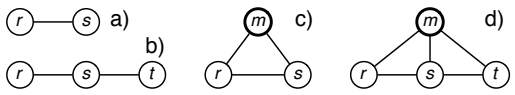

Figure 2: Evolution of Monitoring Schemes

2.3 Neighborhood Discovery

Each node n_i has a neighborhood set N_i which contains its 1-hop and 2-hop neighbors. Let $\mathcal{H}(k, N_i)$ be a function that returns the subset of nodes n_j of N_i with hop count k.

[1]The original watchdog in [7] focussed on omissions.

N_u	
r	1
s	1

N_v	
s	1
t	1

N_m	
r	1
s	1
t	1

Table 1: Neighborhood lists for n_u, n_v and n_m

Thus $\mathcal{H}(1, N_i)$ and $\mathcal{H}(2, N_i)$ represent the sets of 1-hop and 2-hop neighbors of n_i respectively.

Neighborhood discovery is extended from [5]. Note that we do not explicitly describe neighbor authentication, but if desired it can be adapted via the mechanism shown in [5]. In general, when a node v_k wants to join the network it sends a 1-hop HELLO message. Each node v_i receiving this message replies with its 1-hop neighbor list $N_i^1 = \mathcal{H}(1, N_i)$. Now v_k constructs its neighborhood set N_k from those replies it received from n_i during some pre-defined timeout $t_{discovery}$. Thus it adds each n_i as a 1-hop neighbor, and each $n_j \in N_i^1$ as a 2-hop neighbor, unless it is already in N_i as a 1-hop neighbor. This latter case indicates to n_k that n_i and n_j are neighbors themselves. When N_k is assembled, n_k broadcasts it to its neighbors n_i, which in turn have to inform their neighbors about the new 2-hop neighbor n_k.

The 2-hop neighbors are useful when we use the assumptions of [5], i.e., authentication and the assumption that no malicious nodes are present during discovery. However, since we do not limit ourselves to the assumptions of [5], a simple 1-hop discovery actually suffices, i.e., each node v_i that receives a HELLO message from v_j sends a simple reply message. We present the 2-hop discovery only to maintain compatibility with [5] if we should opt to use their assumptions with authentication, and in case we would like nodes to be able to establish neighborhood awareness.

Table 1 shows several table entries for the 1-hop neighborhood discovery in Figure 3. Only direct neighbors, those with distance 1 are in the tables.

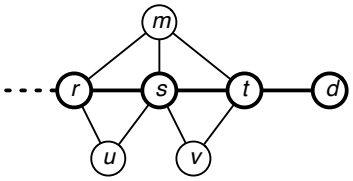

Figure 3: Neighborhood discovery

2.4 Multi-hop Monitoring

Monitor nodes monitor other nodes in a 1-hop or 2-hop fashion. How does a node know what to monitor and what to make of what it sees? A monitor "sees" the world though what it hears, i.e., from messages it hears from its direct (1-hop) neighbors. It may be topology-aware, in which case it can relate what it hears, or topology-unaware. A monitor can operate in a *passive* or *active* mode. A passive monitor just observes, but does not take action. An active monitor reacts by sending notification messages. We will now present several scenarios using Figure 4. For simplicity sake only node indices are used.

Example 1 – omission at node $r1$: Assume that $r1$ fails to forward a packet from q to $s1$. The monitors for $r1$'s behavior are $q, r2$ and $r3$. Node q detects the omission using

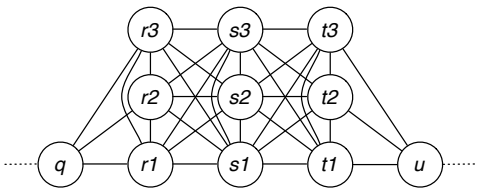

Figure 4: 2-hop monitoring

the monitoring primitive of Figure 2a. It would have seen the message forwarded out of $r1$. Similarly, $r2$ and $r3$ notice the omission using the 1-hop primitive of Figure 2c. Now one monitor will be the first to become an active monitor by informing its neighbors of r's omission via a broadcast message. If this was q then $msg_q = (r$ *did not forward packet with* ID) would be broadcast. Note that q does not need confirmation about what it saw, it witnessed the omission first hand, and could initiate a resend. Now $r1$ or $r2$ will become active first and forward the packet to s. The other will hear this and stay passive. Faults: $\{o\}$ was detected and recovered.

Example 2 – manipulation at node $r1$: This case is similar to Example 1. Nodes $q, r2$ and $r3$ all detect the manipulation (value fault), and $r1$ and $r2$ both send the correct packet to $s1$. Receiving multiple packets, $s1$ uses a majority vote and keeps the correct one. Since q gets the same messages it is aware of the two corrections of the other monitors. Note that the clique $r1, r2, r3$ outvoted the value fault. Faults: $\{m\}$ was detected and recovered.

Example 3 – manipulation at node $r1$*, collaborator at* $s2$: Just as in Example 2 monitors $r2$ and $r3$ detect the manipulation and initiate mitigation by forwarding the correct packet to $s1$. However, now $s2$ may collude with $r1$ sending a false mitigation message with $r1$'s manipulated packet. Now $s1$ gets two correct and two identically manipulated versions and voting does not resolve the problem. Faults: $\{m\}$ was detected at q but no forward recovery is possible.

The problem exposed in Example 3 is that two malicious nodes collaborated, $r1$ was active faulty and $s2$ was passive. Whereas the act of an active faulty node can be observed by monitors, passive framing leaves no detectable trace. Remember that in topology-unaware monitoring a node does not have a complete picture of its neighbor's monitoring capabilities, e.g., $s1$ does not know that $s2$ could not be a monitor for $r1$'s manipulation. This complicates matters. All a node can do is to use a voting function, e.g., majority, to mitigate against conflicting information. However, the threshold for voting in Example 3 was too high.

2.5 Neighborhood Thresholds

The examples of the previous subsection showed the importance of voting and the impact of the threshold of good and bad nodes. Wireless communication cannot produce asymmetric faults within the broadcast domain. Thus simple majority can be used to mask faults, e.g., $N = 2e + 1$ is the number of nodes needed to mask e malicious nodes. Different threshold values for e can be used to deal with malicious nodes depending on the presence and the number of active and passive nodes assumed in the neighborhood.

If we view Figure 4 as a join graph we get Figure 5. Informally, a join graph is defined as a collection of neighboring cliques C_i, where every vertex in the left clique C_i is connected to every vertex in the right clique C_{i+1}. The cliques

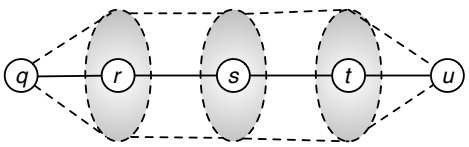

Figure 5: Join graph for Figure 4

are shown as shaded ovals, the join operation is indicated by dotted lines. See [6] for a formal definition of join graphs.

The mechanism for fault detection and mitigation differ if nodes are topology-aware or topology-unaware. Furthermore, in reference to Figure 5 it matters in which clique the faulty nodes are. For example, an observable fault in C_r will initiate the responses of all of its monitors. Passive faults in other cliques, e.g., C_s, will not initiate such responses. Therefore, the impact of these passive faulty nodes has to be masked by the active monitors in C_r.

Now the following can be said about monitoring and recovery from $\mathcal{F} = \{o, d, r, f, m\}$. We use Figure 4 for specific examples and Figure 5 for the general case:

1. $\mathcal{F} = \{m\}$: Assume the only faults are in C_r. Then a node $s \in C_s$ can recover if it receives $N > e$ identical notifications from clique C_r. This is true for the topology-aware and also for tolopogy-unaware cases, since with no colluders outside of C_r there will not be any additional notifications.

   An example of this scenario is $r1$ in Figure 4 as the sole malicious node.

2. $\mathcal{F} = \{m\}$: Assume there are e_r faults in C_r and e_s, e_t passive colluders in C_s and C_t respectively. In the topology-aware case $s \in C_s$ needs to deal only with faults in C_r, as the others can be ignored. Thus s can recover if it receives $N > e$ identical notifications from clique C_r.

   In the topology-unaware case the passive colluders have to be considered, since s does not know which clique nodes belong to. Therefore, it is the burden of C_r to produce enough notifications to compensate for the notifications for the e_s+e_t colluders. This is only guaranteed if C_r is of size $N > 2e$, with $e = e_r + e_s + e_t$, i.e., at least $e + 1$ monitors in C_r respond.

   Example 3 in subsection 2.4 describes such a case.

3. $\mathcal{F} = \{m\}$: Assume scenario 2 above and assume that the colluder in C_s is on the forwarding path. An example of this scenario is $r1$ in Figure 4 as a malicious node and $s1$ as its colluder. As in scenario 2 monitors in C_r provide a majority of notifications about $r1$'s manipulated packet to $s1$. However, just like $s1$ all monitors in C_s received the same majority of notifications and thus have the right packet. Therefore, they will be able to detect if $s1$ colludes by forwarding $r1$'s manipulated packet. This triggers all monitors in C_s to notify $t1$, which is also received by all nodes in C_t.

   Thus, each clique has to resolve the impact of malicious nodes via the respective threshold described in scenario 2 for topology-aware and unaware monitoring. The only difference is the notification scheme that allows forwarding the correct packet into C_t.

Due to spatial limitations we cannot present the scenarios for the remaining faults $\mathcal{F} = \{o, d, r, f\}$. It should be noted however that the cases of $\{o\}$ are special cases of $\{m\}$, i.e., an omission is a special case of a manipulation. Similarly, the cases of $\{o\}$ are special cases of $\{d\}$, which have higher buffer demand. As for fabrications $\{f\}$ at a node, this is detected by the monitor, since the outgoing packet does not have a corresponding packet in the monitors buffer. Misrouting attack $\{r\}$ can be detected by the nature of 2-hop monitoring, since the monitor sees all nodes involved in the forwarding.

3. CONCLUSIONS

This research presented k-hop monitoring as a possible solution to most common packet manipulations in ad hoc networks. The key issue is that no assumptions are made about the behavior of malicious nodes and possible collusions. The monitoring scheme depends on the type of faults considered, which also dictate the thresholds of monitors necessary for attack detection and survivability for topology-aware and topology-unaware implementations. Current focus is on the analysis of the overhead associated with different faults.

4. REFERENCES

[1] M.H. Azadmanesh, and R.M. Kieckhafer, *Exploiting Omissive Faults in Synchronous Approximate Agreement*, IEEE Trans. Computers, 49(10), pp. 1031-1042, Oct. 2000.

[2] S. Buchegger, et.al., *A test-bed for misbehavior detection in mobile ad-hoc networks - how much can watchdogs really do?*, Sixth IEEE Workshop on Mobile Computing Systems and Applications, WMCSA 2004, pp. 102-111, 2-3 Dec. 2004.

[3] C. Chigan, and R. Bandaru, *Secure node misbehaviors in mobile ad hoc networks*, IEEE 60th Vehicular Technology Conference, VTC2004, Vol. 7, pp. 4730-4734, 26-29 Sept. 2004.

[4] Lei Huang, and Lixiang Liu, *Extended Watchdog Mechanism for Wireless Sensor Networks*, Journal of Information and Computing Science, Vol. 3, No. 1, pp. 39-48, 2008.

[5] Issa Khalil, Saurabh Bagchi, Cristina N. Rotaru, Ness B. Shroff, *UNMASK: Utilizing neighbor monitoring for attack mitigation in multihop wireless sensor networks*, Ad Hoc Networks, Vol. 8, pp. 148-164, 2010.

[6] A. Krings, and Z. Ma, *Fault-Models in Wireless Communication: Towards Survivable Ad Hoc Networks*, Military Communications Conference, MILCOM 2006, pp. 1-7, 23-25 Oct. 2006.

[7] Sergio Marti, T. J. Giuli, Kevin Lai and Mary Baker, *Mitigating routing misbehavior in mobile ad hoc networks*, Mobile Computing and Networking, pp. 255-265, 2000.

[8] A. Patcha, and A. Mishra, *Collaborative security architecture for black hole attack prevention in mobile ad hoc networks*, Proc. Radio and Wireless Conference, RAWCON '03, pp. 75-78, 10-13 Aug. 2003.

[9] N. Song, L. Qian, and X. Li, *Wormhole attacks detection in wireless ad hoc networks: a statistical analysis approach*, Proc. 19th IEEE Intl. Parallel and Distributed Processing Symposium, 8 pages, 4-8 April, 2000.