RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT
INSTITUT FÜR INFORMATIK 4

Emulative Evaluation of a Multimodal Approach for Detecting Routing Attacks in MANETs

Bonn, 28th July 2011

Author:
Jonathan P. Chapman

Advisor:
Prof. Dr. Peter Martini
Second Advisor:
Dr. Michael Meier

Er hat zu viel Geist, er fährt mit seinem Geist wie auf einem Zauberwagen über die Erde, auch dort, wo keine Wege sind. Und kann es von sich selbst nicht erfahren, daß dort keine Wege sind. Dadurch wird seine demütige Bitte um Nachfolge zur Tyrannei und sein ehrlicher Glaube, „auf dem Wege“ zu sein, zum Hochmut.

Dr. jur. Franz Kafka, viertes Oktavheft

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit selbstständig erstellt zu haben. Alle Zitate sind als solche gekennzeichnet und ich habe keine anderen Hilfsmittel und Quellen als die angegebenen verwendet.
Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Jonathan Pascal Chapman
Bonn den 28. Juli 2011

Contents

List of Figures

List of Tables

Glossary

ETX Expected Transmission Count. 4

FN Fully-equipped Node. 13

IDS Intrusion Detection System. 9

LN Lightweight Node. 13

MANET Mobiles Ad-hoc Network. 3, 9, 11, 13
ME MotionEmulator^{NG}. 4
MES MotionEmulator^{NG} Server. 4, 5
MPR multipoint relay. 3

NTS Nettalker Server. 9

OLSR Optimized Link State Routing. 3, 11
OLSRd OLSR daemon. 3, 4, 9

RITA Responsive Intrusion detection for Tactical Ad hoc networks. 4, 9, 13

TC Topology Control. 3
ToGBAD Topology Graph Based Anomaly Detection. 9–11

VE virtual environment. 13

Chapter 1

Introduction

In May 2011, United States officials announced that cyber attacks against them may constitute an act of war and will be met with the corresponding response, including the option of conventional military action [57]. While igniting a debate about whether or under which conditions armed responses may be justified, the declaration was not a complete surprise. As early as 2003, a revised version of the United States Army Field Manual FM-3-13 on "Information Operations" was adopted and approved for public release [60], introducing cyber operations and the term Computer Network Operations (CNO), subdivided into Computer Network Attack (CNA) and Computer Network Defense (CND). In 2009, the United States Strategic Command was ordered to establishment a Cyber Command, which was declared to have achieved intial operational capabilities less than a year later [50] and finally in late 2010, Deputy Secretary of Defense William J. Lynn further emphasised the significance of cyber space in the Department of Defense's planning by declaring it a domain of warfare on its own [52].

The aforementioned announcement might be seen as the result of a 2009 report compiled by the RAND Corporation and sponsored by the United States Air Force [42]. In "Cyberdeterrence and Cyberwarfare", the author not only discusses questions concerning strategic cyber warfare but also explicitly mentions the option of deterring attackers by using the threat of conventional warfare. However, in contrast to this wide interpretation of the term and measurements, the military sources mentioned above indicate a more narrowly focused picture of cyber warfare. [60] focuses on the idea of using cyber, but also conventional, capabilities for degrading, exploiting or influencing enemy communications to affect their ability to make informed decisions on the one hand and for protecting one's own resources against such attacks on the other hand. The public mission statement of the Cyber Command even indicates an emphasis on the latter, suggesting that even within the militaries supposedly best prepared for cyber warfare, the debate is still open as to what actions may actually constitute such a war.

[61], a study on cyber security risks conducted for the Organisation for Economic Co-operation and Development (OECD), points out the significance of the Internet and services connected to or relying on it for the OECD member countries' economies and societies. While they conclude that the security of these systems is thus of utmost importance, they express scepticism about the perspective that there may ever be a conflict deserving the term "cyber war". Bruce Schneier shares this view [58], later adding that the term war may be a particularly bad choice for describing the potential challenges faced through information technology infrastructure and bearing the potential to lead to wrong conclusions [59].

Removed for Copyright Reasons

Removed for Copyright Reasons

Figure 1.1.: Left: A soldier accessing a map on the display of the group leader equipment ("Display tragbarer Führungsrechner") provided through the IdZ-ES programme (Source: Bundeswehr/Rott).
Right: A soldier communicating through a current issue platoon radio during a mine clearing operation in Char Darrah district, northern Afghanistan (Source: Bundeswehr/Kaiser).

Despite the confusion about the proper scope of the term cyber warfare, there is little doubt that one act, although reports were never confirmed by either of the parties involved, would qualify even under the narrowest definition. In 2007, Israeli planes bombed a suspected nuclear facility in northern Syria, allegedly after executing a CNA against the Syrian air defence systems that allowed their planes to enter Syrian airspace undetected [26]. While one might argue that this action presumably saved lives on the part of the Syrian air defence crews, it deprived them from of the ability to perform their mission, i.e. to defend Syrian airspace, rendering the suspected operation a perfect example of Information Operations in the sense of [60].

This example makes clear that it will not be enough to build and maintain communication infrastructures for 21st century warfare, but that militaries adapting their tactics and training to these capabilities, as pointed out e.g. in [7], also have to establish proper means for defending them. Militaries that fail to do so may be unable to leverage their technological advantage or even suffer disadvantages when facing an enemy determined to and capable of attacking these infrastructures. Figure 1.1 shows an Infanterist der Zukunft-Erweitertes System (IdZ-ES) handheld device on the left, demonstrating the German army's vision for a MANET based communication device which would replace current radio equipment such as the backpack sized platoon radio displayed on the right hand side of Figure 1.1.

In this thesis, we will consider two particular kinds of attacks against tactical MANETs, the kind of network that will most likely be predominant close to the leaves of a future military's or emergency service's communication infrastructure. First, we want to implement the sinkhole and wormhole routing attacks in order to be able to determine their potential impact if deployed against a MANET ill-prepared to cope with them. Secondly, we want to evaluate and improve a set of methods for detecting these attacks in tactical MANETs.

This thesis is structured as follows: We start with a short introduction to the prerequisites for this thesis in chapter 2. In the following chapter, we briefly discuss the related literature, providing an overview to current and past approaches in the field. Chapter 4 describes the Topology Graph Based Anomaly Detector (ToGBAD) approaches for detecting routing attacks in MANETs. In the next chapter, we describe our environment for evaluating the impact of the named attacks and ToGBAD's ability detect them. Following in Chapter 6, we describe our implementations of the attacks and their impact in our evaluations and Chapter 7 presents our findings concerning ToGBAD's ability to detect them. This thesis closes with our conclusions and a brief summary in Chapter 8.

Chapter 2

Prerequisites

2.1. Optimized Link State Routing (OLSR) and OLSRd

Route discovery in MANETs falls into two major categories; reactive protocols establish routes as they are required, e.g. by applications trying to send a packet to a node without a known route, while proactive protocols maintain routes to all or some nodes in the network even when there is no data to deliver to these nodes. The Optimized Link State Routing (OLSR) protocol, standardised in RFC 3626 [15], falls into the second category and ports the link-state routing paradigm to MANETs.

The mechanisms used by OLSR can be divided into two major categories, neighbourhood discovery and topology discovery. OLSR uses regularly emitted HELLO messages to achieve the former. They will be broadcast on the link layer and not forwarded, thus a node receiving another node's HELLO messages knows that it can receive packets sent by that node and includes this information in its own HELLO messages. This allows nodes to discover not only their immediate neighbourhood, but also their two-hop neighbourhood. They use this information to select a subset of their neighbours, the multipoint relays (MPRs), with a combined neighbourhood that would cover all the two-hop neighbours of the MPR selector.

Since the neighbourhood discovery process described in the previous paragraph is limited to providing information about nodes within a distance of at most two hops from a node, the topology discovery mechanism provides information learned through that mechanism to the other nodes in the network. OLSR nodes will regularly emit Topology Control (TC) messages containing at least the addresses of nodes that selected them as an MPR. Other than HELLO messages, each unique TC message will be forwarded by each MPR selected by the node generating or forwarding the message. Since the MPRs are chosen to cover all two hop neighbours of a given node and any other one hop neighbours are expected to receive any transmission anyway, TC are expected to be received by all nodes in the network. Also, the selection of MPRs implies that each node in the network should be included in at least one other node's TC messages. This allows nodes to construct a graph from the information extracted from all received TC messages that would include all nodes in the network. OLSR will determine routes in this graph by choosing the shortest path in terms of hops to each node in the graph and include the next hop on these paths as the gateway to the respective target in the routing table of the node running OLSR.

The OLSR daemon (OLSRd) is an implementation of the OLSR protocol which is available on several platforms. While originally written as part of an M.Sc. thesis, it was released under an open source license and was subsequently improved and extended by volunteers. It has been fielded in several city wide mesh networks, such as Freifunk [25] and FunkFeuer [4]. As early as 2004, just about a year after the first public release of the OLSRd, support for link quality metrics was added on the basis that the hop count metric implied by RFC 3626 would maximise the geographical distance covered by each transmission. While this would be favourable in wired networks, where transmission errors are rare and are seldom related to the geographical distance between the peers, the received signal strength drops at a cubic rate with respect to the distance in MANETs and transmissions are thus more likely to fail when covering large distances. While the use of a link quality metric breaks the OLSRd's compliancy with RFC 3626, it is generally considered to be RFC compliant when run with an appropriate configuration.

2.2. Expected Transmission Count (ETX)

Since discovering routes was a well-researched topic in wired networks, when wireless ad-hoc networks were surfacing first approaches to determine routes in these kinds of networks were based on the shortest path first paradigm found in many protocols for routing in wired networks. The implication that covering larger geographic distances with each transmission is beneficial turned out to be disadvantageous in wireless networks where the strength at which signals are received is strongly related to the distance a signal had to travel and where interference is much more common due to the medium being shared by all nodes operating at a given geographical location.

This discovery lead to the introduction of link quality metrics as a way to meter the cost of routes not by the number of hops that had to be traversed but by the cost of delivering a packet on a given route. Since a metric accurately reflecting the properties of the modelled channel provides a better base for routing decisions, routing mechanisms considering link quality information generally provide a higher throughput and lower packet loss rates than their counterparts. In wireless networks, common link layers such as IEEE 802.11 [1] initiate a limited number of retransmissions for each unacknowledged unicast transmission. Thus, while packets travelling along a path with low link qualities or high packet loss ratios may often still be delivered through retransmissions by the link layer protocol, the channel occupancy and energy consumption required for delivering a packet through such a route may be several fold of that required for a path traversing more hops but travelling through links with higher link qualities.

The most prominent link quality metric for wireless networks is the Expected Transmission Count (ETX) metric first proposed in [**?**]. Their goal was to improve the throughput obtained in a wireless network by reducing the amount of retransmissions on a selected path which they achieved by introducing their metric in the routing process. Since retransmits will occur not only when delivery of a given packet fails but also when the delivery of an acknowledgement for that packet fails, their metric considers the quality of both unidirectional links between two given nodes. A unidirectional link's quality will be determined by observing the count of packets received through that link and comparing it against the count of packets expected to be received in a given time span. To combine the packet delivery ratio (PDR) achieved on a link from a node to its neighbour, the forward link quality or d_f, and the ratio d_r achieved on the reverse link, the ETX metric uses the following formula, defined as the expected number of transmissions required for delivering a packet through a given bidirectional link:

$$ETX = \frac{1}{d_f \times d_r}$$

With $d_f \to 0$ or $d_r \to 0$ this equation results in infinity, following our intuition that if at least one link does not deliver any packets, transmitting packets through the respective bidirectional link will never result in a successful transmission. Perfect links will have a packet delivery ratio of 1 and result in a single transmission while less than perfect links will yield an ETX value larger than 1. Since delivering a packet through a given path requires that the source and each intermediate node transmit it successfully to the next node on the path, the cost for a path can be obtained by adding the ETX values of all links traversed.

2.3. MANET Routing Attacks

2.3.1. The Sinkhole Attack in MANETs

The sinkhole attack is an attack where a node tries to manipulate the routing in a network to increase the share of traffic being routed through the attacker. This can serve several purposes; the most common one is to execute a denial of service attack by attracting routes but silently discarding all packets that were supposed to be forwarded by the attacker. This is often called a blackhole attack in MANET contexts. In a variation of this attack, an attacker would selectively drop packets, e.g. depending on their content or a random pattern. This variation is sometimes called a greyhole attack, referring to the somewhat restricted blackhole behaviour of such an attacker. It is noteworthy that a pure sinkhole attack may be limited to attracting routes while not engaging in any obviously destructive behaviour as the previously described variations do. However, depending on the routing mechanism, the attacker may be limited in his ability to forward packets as a consequence of the success of its own attack.

With the diversity of the mechanisms employed in routing protocols comes an equally large diversity in possible implementations of the sinkhole attack. Two general approaches are to propagate unfavourable routing information, spoofing the sender address, to divert routes from other nodes or to propagate favourable routing information with regard to oneself to attract routes. Since the former approach is rendered impossible through the employment of cryptographic authentication and integrity checks, we focus on the latter which only requires that the attacker possesses valid key material for itself. We would also like to point out that some publications, e.g. [47], use the term blackhole when referring to any of the variations described in the previous paragraph. In this work, we use the term sinkhole when referring to the general concept, including all of its variations. When referring to a specific variant, we will use respective term, i.e. blackhole or greyhole.

Since our work focuses on the OLSR routing protocol, we provide a brief description of the sinkhole attack in OLSR based networks. In such a network, an attacker would include nodes in its HELLO messages that are not among its actual neighbours. The real neighbours would in turn consider the forged neighbours to be two-hop neighbours that could be reached through the attacker. Figure 2.1 shows the topology of a small network of six legitimate nodes (Ⓐ through Ⓕ) and one executing a sinkhole attack (node Ⓢ). Smaller shaded circles attached to

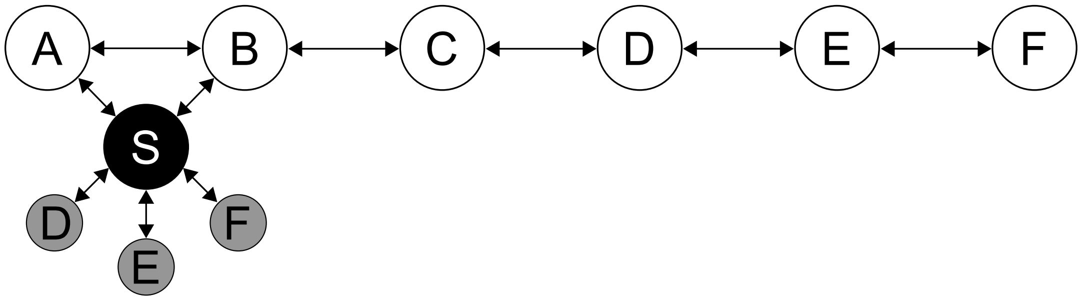


Figure 2.1.: Schematics of a sinkhole attack in OLSR. Nodes Ⓐ to Ⓕ are legitimate nodes, Ⓢ is perpetrating a sinkhole attack by claiming to be a neighbour of nodes Ⓓ through Ⓕ (shaded nodes) while it is in fact only a neighbour to nodes Ⓐ and Ⓑ.

Ⓢ indicate neighbourships forged by that node. Since the routes through Ⓢ are more attractive than their legitimate counterparts, with the exception of the route from Ⓑ to Ⓓ which is equally attractive, if the standard hop count metric is used, Ⓢ would succeed in affecting the routing of its two neighbours, Ⓐ and Ⓑ.

Since HELLO messages are never forwarded, the influence of Ⓢ's attack is limited to its actual neighbourhood. While the attacker could decide to forge TC messages as well to extend the scope of its attack, doing so would simplify its detection since eventually all nodes, including those which the attacker forged neighbourships to, would receive the TC message. In the scenario presented in figure 2.1, the only potential for increasing the success of Ⓢ's attack is node Ⓒ where the claimed three hop route through Ⓑ and Ⓢ would compete with a legitimate three hop route through Ⓓ and Ⓔ.

2.3.2. Link Quality Forgery in MANETs

While not qualifying as a routing attack by itself, forging link qualities is a method that can be leveraged by other routing attacks in MANETs that use a routing protocol with link quality support. Since such a protocol will prefer links or routes with lower cost or higher quality links, an attacker can use forged link quality values to support its attack. Such use may include:

- Decreasing the link quality to reduce the likelihood of being selected as a hop on a route (node misbehaviour)
- Increasing the advertised link quality for existing or forged links to support sinkhole attacks
- Advertising low link qualities using the ID of another node to reduce the likelihood of that node being chosen during route set-up to support sinkhole attacks
- Advertising high link qualities using the ID of another node to increase the likelihood of that node being chosen as a hop on a route (to support node misbehaviour or when collaborating with a sinkhole attacker)

Note that the second attack, increasing advertised link qualities, is the most likely one since it does not require any key material that would not usually be available at a given legitimate node. In the example for the sinkhole attack described in Section 2.3.1, link quality forgery could enable the attacker to extend the scope of its attack. Assume that all links in the network, including the links forged by Ⓢ have a cost of 1 except for the link between nodes Ⓔ and Ⓕ,

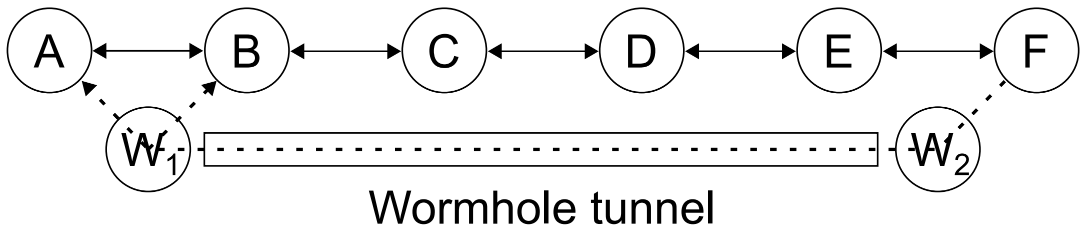


Figure 2.2.: Schematics of a wormhole attack. Nodes Ⓐ to Ⓕ are legitimate nodes, Ⓦ$_1$ and Ⓦ$_2$ wormhole endpoints. The dashed line indicates the path of a frame captured and forwarded through the wormhole tunnel by node Ⓦ$_2$ and then being retransmitted by Ⓦ$_1$.

having a cost of 10. Upon receiving a TC message including Ⓢ's forged neighbourship to Ⓕ, Ⓔ would calculate the combined cost of the links traversed to Ⓕ when routing through Ⓢ as $5 \cdot 1$ and compare it against the cost of 10 for delivering a frame through its own link to Ⓕ. Since the cost of the route through the attacker would be significantly lower, Ⓔ would choose the route through Ⓢ.

2.3.3. The Wormhole Attack in MANETs

A wormhole attack is perpetrated by two or more collaborating nodes in a wireless network. Each of the attackers captures wireless transmissions and transfers the captured frames to its collaborators through a tunnel. Upon receiving such a forwarded frame, an attacker will retransmit it on the attacked wireless channel, creating the illusion of locality between the capturing node's and the retransmitting node's neighbours where in fact there is none.

While the attack in its basic form does not degrade the performance of the attacked network or may even serve to enhance it, it is likely to affect the routing in the attacked network. Frames transmitted by the attacker's neighbours and amplified by the wormhole attack cover unusual geographic distances and thus often provide a much cheaper alternative to legitimate routes not only for the attacker's immediate neighbourhoods but also their wider surroundings. Thus, packets travelling between these surroundings are likely to travel through the wormhole link.

Attackers may exploit this property in several ways. First of all, it allows them to eavesdrop on a larger portion of the attacked network's traffic than they could by simply passively eavesdropping on transmissions in their reception range. While encryption may prevent the attacker from understanding the contents of the captured packets, traffic analysis may still reveal hierarchies between nodes, hardware, operating systems (using programmes such as Nmap [43] or p0f [69]) or protocols in use (cf. [16]) or in some case even allow guessing their content (cf. [68]). However, by providing an attractive connection which is under their complete control, the attackers may also decide to selectively discard frames, degrading or even disrupting the connectivity between the affected segments of the network. Finally, if packets are not adequately protected using cryptographic methods or the attackers have access to valid key material, they may choose to modify or inject packets to confuse or mislead the attacked nodes or their users.

Wormholes are generally distinguished by the nature of the tunnel they use for forwarding captured frames between the participating nodes. In-band wormholes encapsulate the captured frames and forward them through the same network targeted by their attack while out-of-band wormholes use a second channel, such as a stronger radio or even a wired link, for that task. It is noteworthy that in-band wormholes not only suffer significant delays between capturing and retransmitting a frame since it has to be forwarded, probably among several hops, through the attacked wireless network, but also require a node relaying their packets to avoid falling victim to the success of their own attack. [44] analyses the requirements for placing such a relay to allow a wormhole to be sustained, however the requirements appear to be too tight to allow an in-band-wormhole attack to work in practice. Thus, unless otherwise stated, this thesis refers to out-of-band wormholes.

In addition to the types of wormhole attacks discussed in the previous paragraph, [37] discusses further refinements of the wormhole concept. These includes single node wormholes, where a single note retransmits all frames it received, and wormholes created through colluding nodes claiming neighbourships with each other while in fact they are not neighbours. Note that these nodes should still possess a means to forward frames between each other since otherwise this would only qualify as a variation of the sinkhole attack described in Section 2.3.1.

Chapter 3

Related Work

In this chapter, we provide an overview of the literature on the subjects covered in this thesis. Section 3.1 provides an overview of approaches suggested for detecting sinkhole attacks in MANETs. As for the approaches for detecting wormhole attacks described in Section 3.3, the variety of approaches and the volume of respective publications prohibits an exhaustive discussion of the relevant literature. We thus provide an overview of a mix of particularly prominent, recent or intriguing approaches discussed in the scientific community. As opposed to the sinkhole and wormhole attacks, little research exists concerning link quality forgery. Section 3.2 thus provides a more in-depth discussion of the two publications concerned with detecting forged link qualities. This chapter closes with a brief summary in Section 3.4.

3.1. Detection of Sinkhole Attacks in MANETs

Sinkhole attacks are among the attacks that have received the most attention among researchers in the MANET and wireless sensor networking communities. We thus do not provide an exhaustive overview of the related work but point out examples of the most prominent approaches. The methods we present here are thus illustrative of the approaches we encountered most often during our research, however in some cases we have chosen deliberately to present earlier versions that are more appropriate for describing the principal concept. Additionally, where indicated we deviate from this principal and present some concepts which were particularly creative or caught or attention for other reasons.

Sinkholes were one of the target of the approach described in [45], one of the earliest works in MANET intrusion detection. They use two components, a component called watchdog sets the wireless Network Interface Controller (NIC) to promiscuous mode and listens for retransmissions of packets that the node running the watchdog sent to a neighbour for forwarding. If the expected retransmit did not occur within a specified period, it increases a counter indicating that the node failed to forward the packet. When the fraction of packets that were not forwarded by a given node exceeds a threshold, the node is considered to be perpetrating an attack. This information is used by the second component, the pathrater, to avoid setting up routes through nodes which appear to be executing an attack.

As the authors of [45] already pointed out, their approach exhibits several limitations. First of all, if the MANET does not employ a source routing based protocol, the watchdog cannot verify whether the next hop forwards the packet to the correct node, i.e. the use of this approach limits the choice in routing protocols. Even if source routing is used, the mapping between link layer and routing layer addresses must be fixed and available to the watchdog, otherwise an attacker could simply retransmit the packet using a random link layer destination address. While this may be too expensive for a selfish node that tries to conserve battery power by not forwarding packets, a malicious attacker would exploit this to be able to execute its attack without being detected. Thus, we consider the approach not practical for the kind of networks we consider.

The basic concept is however still present in many reputation based schemes including recent ones such as the one presented in [46]. As in [45], nodes monitor whether their neighbours appear to forward packets relayed to them, thus the approach suffers from the same deficiencies as the approach described above.

In [18] the authors describe their own Ad hoc On-Demand Distance Vector (AODV)-like MANET routing protocol which was designed to include authentication of routing messages as well as switching between the best and second best route discovered. While custom routing protocols are commonly found in sinkhole detection schemes, the latter feature is unique to this approach. The authors assume that it is easy to forge the best route but claim that forging the second best route is more difficult. An attacker that is able to forge the best route may however safely assume that by increasing the cost of the claimed best route by the lowest non-zero value yields a good chance of being accepted as a second best route. Apart from a lack of a practical implementation of the protocol, the introduced mechanisms require a source routing protocol and thus institute tight constraints on what a resulting routing protocol might look like.

The ADVSIG protocol, extending OLSR, described in [55] also includes signatures in routing messages, allowing nodes to authenticate their content. Other than the protocol proposed in [18], nodes however not only include signatures on their own messages, but also individual signatures which they call certificate and proof. The earlier are included in HELLO messages and are signatures on the address of a neighbour, the timestamp of generating the signature and the link state at that time. A neighbour of a node receiving a certificate referring to the state of its link to the given node may include that certificate in its own HELLO and TC messages through which it becomes a proof. Since asymmetric signatures are being used, any node may check these proofs to verify whether the claimed status is valid given the proof provided. Note that both of the approaches presented last require an extensive number of digital signatures to be created and verified, in the case of ADVSIG not even one signature per routing message but a number of signatures in the order of the average neighbour degree in a network has to be verified for each message received. This appears inappropriate for a MANET environment where devices may have to observe tight limits in processing and battery power.

Another approach to establish secure routes is to use signatures identifying either a normal behaviour or attacks by comparing observations with a predefined pattern. An example for such a mechanism for detecting sinkholes in networks using the Dynamic Source Routing (DSR) protocol can be found in [17]. While the rules they describe appear valid, two of them are derived from observations in simulations, i.e. need further verification. Also, as all signature based approaches, it is intrinsically tied to the protocol the signatures were defined for and can thus not be adapted for other protocols. A signature based approach for OLSR is described in [66]. The defined rule set primarily relates TC messages with information learned from HELLO messages, limiting this approach to detecting forged TC messages while forged HELLO messages would remain undetected. [51] closes this gap by describing signatures that allow a node to validate HELLOs received with entries referring to itself. Their second set of rules checks whether a neighbourhood sensing appears to be legitimate in regard to another node, however rules may be violated if broadcasted packets are not received by the detecting node. This illustrates the limits of signatures based on protocol behaviour; while proper protocol behaviour can be formalised, a node which is not actively engaged in it may not be able to receive each transmission and thus signature mismatches are very likely to occur. A signature based approach would thus either suffer a significant amount of false positives or require a supplement that mitigates this problem. However none of the publications on signature based approaches for OLSR we are aware of describes such a mechanism.

[56] presents an acknowledgement (ACK) scheme where nodes sending data inject ACK-requests in their packet streams which would be answered with response by the destination. If the ratio of responses received drops below a threshold, the approach would classify the route to the destination as being under attack. However the authors fail to require authentication even of ACKs but assume that the attacker is not aware of how to generate responses, which could be formulated as a violation of Kerckhoffs's principle which states that a system should be secure even if the attacker knows everything except the keys. In the next step after detecting an attacker, their algorithm will inform the other nodes in the network and then initiate a new route discovery. The authors of [56] state that this improves the PDR in a network where 50% are attackers, indicating that their approach does not provide an benefit in the presence of more reasonable ratios of attackers.

[6] presents a more recent ACK-based approach, a refinement to the Two-ACK scheme for detecting malicious packet dropping. Two-ACK prevents intermediate nodes from silently discarding packets by each node sending an ACK for each forwarded packet to the second last hop on the route. The refinement reduces the overhead, letting only the destination node send an ACK if however the source does not receive an ACK for a packet, it switches to the Two-ACK scheme for the next packets it wants to sent to the given destination. Since an attacker could simply forge the ACK, this approach again only makes sense if the authenticity and integrity of messages can be verified. While the end-to-end ACK mechanism should work regardless of the routing protocol employed, the Two-ACK scheme requires a source routing protocol, i.e. nodes must be able to determine the second next hop on the route to be able to verify the authenticity of an ACK. The scheme is however limited to detecting packet drops, it cannot determine whether or not a route is legitimate or has been influenced by a sinkhole attack. Finally, the overhead created by the scheme is indicated to be between 20% and 40% in scenarios with what is called a high degree of movement. Since this affects not only the channel capacity but also the resource consumption of battery powered devices, this appears prohibitively high for MANET environments.

[63] takes a unique approach against the sinkhole attack by introducing randomisation into the routing process of the AODV routing protocol. While their results suggest that this approach improves packet delivery ratios in the presence of an attacker which is located close to the source node, the average end-to-end delay increases by about eight to 15 fold, indicating that the efficiency of the route discovery process is strongly affected by their approach. Introducing randomisation into the routing process thus appears ill-advised.

3.2. Detection of Link Quality Forgery in MANETs

As of we know only few approaches exists for detecting link quality forgery in wireless networks so far. [13] describes such an approach for wireless sensor networks. It assumes that the network consists of sensor nodes and a few detector nodes with more powerful batteries and an exclusive communication channel, nodes should not move at all. Routing will be done with an on demand distance vector protocol using route requests (RREQs) and route replys (RREPs) including a link quality indicator. The authors argue that since there is no mobility, nodes are able to determine and store the minimum cost for the links to each of their neighbours. Additionally, detector nodes will calculate the cost for the cheapest route to the closest other detector nodes.

Note that if nodes are able to move independently of each other, their relative distance may change during the lifetime of a measurement. Since the free space model already implies that the receiver signal strength drops at a quadratic rate in relation to the distance covered by the signal, node movement may have a drastic impact on the signal strength and thus the quality of a link between given nodes so that it may be completely unrelated to a previously determined link quality. Thus, we have to point out that the assumption of being able to determine minimum link qualities does certainly not hold when nodes are moving independently of each other.

The approach described in [13] now introduces two methods for detecting forged link qualities. First, each node checks the link qualities advertised by its neighbours against the stored minimum cost for the given link multiplied by a tolerance factor, if the advertised link quality falls below that value, the node assumes the link quality value has been forged. Secondly, detector nodes monitor link qualities advertised in RREPs. When a given RREP is captured by several detector nodes, the detectors may determine the cost of the intermediate path for these nodes, i.e. the difference between the observed link qualities. If the cost increase is less than the cost of the routes between the detector nodes closest to the given nodes minus the cost of the links from them to the closer node, they assume that the value has been forged.

While the assumption that nodes will generally be immobile prohibits the use of this protocol in MANETs, where mobility is one of the defining features, there are several other issues discouraging any effort to enhance it appropriately. To name a few, it appears unlikely that minimum link qualities can be determined reliably; even in a static scenario environment conditions such as rain or moving objects may have a very significant impact on the link qualities obtained during any kind of initialisation phase, rendering any decisions based on them invalid. Countering this effect by an appropriately large tolerance factor may allow the attacker to execute an effective attack without triggering the mechanism. Also, the first method assumes that an improved link quality constitutes an attack, specifically even if the link quality is improved by using a stronger radio. Effectively prohibiting the use of stronger radios unless they have been used in the initialisation limits the node's ability to adjust their behaviour to environment conditions, such as an improved power supply or the need to cope with strong interference. Last but not least, the second method requires a significant amount of signalling between the detector nodes for each overheard RREP. They not only have to determine the minimum link quality from themselves to the transmitting node, but also have to discover at least one more detector node that the given RREP passed by and exchange the appropriate information with that node.

The second approach for link forgery detection we are aware of was described in [40] and was also developed for wireless sensor networks. The authors describe countermeasures in two routing protocols for such networks. While we assume that impersonation attacks are not possible in the networks we consider and thus do not take into consideration this part of their approach to detect attacks in networks using the MintRoute protocol, the second part deserves consideration. It is based on the observation that the unidirectional links between two neighbouring nodes are generally subject to the same environmental effects and thus the link qualities determined for them differ only to a small degree, which they observed to be less than 20% of the maximum link quality value.

While the approach appears sound in the type of networks assumed in [40], we have to point that in MANETs nodes may be using different kinds of hardware and power supply, allowing some nodes to use more powerful transmissions to cover larger distances. If however nodes are transmitting at different power levels, this will also affect the quality of the unidirectional links between them, increasing the difference that would be observed for the unidirectional link quality values. Depending on the differences in transmission power that would have to be tolerated, an attacker might be able to adjust his attack without suffering a significant decrease in the effectiveness of his attack.

[40] also describes two approaches for detecting forged link qualities in networks using the MultiHopLQI routing protocol. The protocol assumes that each node chooses a parent node based on the total path cost to the destination, determined by adding up all the inverted link quality values along the route. Thus, the first check proposed is for nodes to listen on their neighbour's route advertisements whether the route cost decreased when compared against the value advertised by the parent. So the check requires not only a node which is able to hear both the parent's and child's advertisements, but is also very tightly linked to the mechanisms of the routing protocol. The second approach resembles the approach discussed in the previous two paragraphs, where nodes assume that the link qualities for the unidirectional links between them should not differ by more than a given value, and suffers from the same drawbacks we discussed there.

We conclude that the approaches for detecting link quality forgery that have been published so far are not appropriate for generic MANET environments. Besides heavily depending on the routing protocol, each approach imposes restrictions such as node immobility or the use of the same signal level for transmissions by all nodes, regardless of their capacities, that are at best undesirable in MANET environments.

3.3. Detection of Wormhole Attacks in MANETs

Another topic that received a great deal of attention is the wormhole attack. It solely requires access to the attacked network layer and since the layer is wireless in the case of MANETs, preventing access is close to impossible. As for the sinkhole attack we abstain from providing an exhaustive overview on the related literature but point out examples for some of the most prominent approaches. As pointed out in Section 2.3.3, we consider in-band wormholes impractical and thus do not consider approaches which only work for in-band wormholes.

Early timing based approaches such as the Temporal Leashes concept described in [34] require very tightly synchronised clocks, which cannot be guaranteed in a MANET. Thus, more recent timing based approaches such as the one presented in [14] use offsets or local timing, only. The protocol described in [14] is based on nodes monitoring their neighbourhood when forwarding RREQs. For each neighbour a timer is started, an attack is assumed to take place when the first node overhears a RREQ being forwarded after the timer for the respective node expired. The formula suggested for determining the time-out is twice the transmission range divided by the signal propagation speed or, when assuming radio signals travelling at the speed of light with a maximum transmission range of 300m, approximately:

$$\frac{2 \times 300m}{2.998 \times 10^8 \frac{m}{s}} \approx \frac{200.133}{10^8} s \approx 2\mu s$$

However in contention based wireless networks, nodes may only start transmissions when the channel has not been occupied for a given duration. In IEEE 802.11 networks the minimum waiting time or Short Interframe Space (SIFS) has a duration of 10μs (in IEEE 802.11b/g) or five times the suggested time-out. However SIFS would generally not even be applicable for transmitting RREQs, i.e. regular IEEE 802.11 layer 2 behaviour would already result in triggering an alarm unless a transmission range of well beyond 1000m is assumed to be acceptable. When considering nodes with more than one neighbour, even without any inter frame spacing and processing time but successful collision avoidance only one node would be able to meet the time-out. While these issues could be mitigated by adapting the threshold respectively, an attacker could retransmit frames immediately, disregarding any inter frame spacing and collision avoidance, to remain below that threshold. Most timing based approaches suffer from similar problems, i.e. they do not consider processing or back off times and are thus not applicable in MANETs employing current technology.

[49] presents a different timing-based approach. Nodes should include their local high precision time in periodically broadcasted reference frames that are related to the routing protocol, e.g. frames containing HELLO messages. These would allow other nodes to determine the offset between their own and the transmitting node's clock. Under the assumption that clocks are not synchronised but no clock drift occurs, these offsets will be constant. Their method indicates that a wormhole attack is being perpetrated, if the difference between the drifts observed by a neighbour exceeds a given threshold. The offsets may however have all been obtained from frames tunnelled through a wormhole, thus, unless the delay introduced by the wormhole is volatile beyond the given threshold, the method will not be able to detect an ongoing attack under these circumstances. We would also like to point out that the authors of [49] used an ns-2 Network Simulator (NS2) implementation the evaluation of their method which adds timestamps to packets generated by the OLSR routing protocol, i.e. at OSI layer 3. While this approach should allow using of-the-shelf hardware even when securing routing messages with signatures, it makes them subject to the second layer effect described in the previous paragraph, since they cannot be modified after they have been submitted to the NIC for transmission. This would leave a real world implementation with either introducing thresholds large enough to compensate for back offs introduced by the layer 2 protocol and thus may not be able to detect an attack, if the attacker retransmits forwarded frames very fast. The alternative of moving the protocol to the second layer would require custom hardware and layer 2 protocols writing and signing timestamps to frames as they are transmitted. Introducing such hardware would however significantly increase the cost of production for each node while still protecting the network against a single type of attack only.

The authors of [24] describe a method for which they took that approach, i.e. they define a modification of the IEEE 802.11 layer 2 protocol which exchanges nonces in the CTS and data frames in an adapted RTS/CTS mode. Since the CTS will time out, if the node initiating the transmission does not send a data packet immediately after a SIFS or 10μs passed, the attacker has to be able to able to capture and retransmit frames in less than 5μs unless it would generate its own CTS and data frames. To prevent the latter, nodes will exchange signatures on both nonces, allowing them to verify that the previous exchange was legitimate. This yields, besides the fact admitted by the authors that a short time frame will exist in which a connection may be considered legitimate even though the nonce exchange has not been authenticated yet, a potential for a significant performance impact. Not only does signature calculation requires asymmetric cryptography which is computationally expensive but the verification has to repeated periodically to ensure that links are still not subject to a wormhole attack. With the suggested time-out of 30 seconds for each verification, a node with 15 neighbours will have to create and verify 15 signatures each for each time frame of 30 seconds, i.e. one asymmetric cryptography operation per second. In addition to that, while the authors claim that it would be sufficient to modify an existing IEEE 802.11 compliant NIC's firmware to implement their protocol, they provide neither such an implementation nor any arguments to back their claim up. Given the obstacles that would have to be overcome by any of the timing-based approaches considered above before being eligible for a practical implementation, it appears unlikely that a timing-based approach will provide a means for properly distinguishing non-attacked networks from such being attacked with a wormhole.

In [31] an approach is presented that analyses the time between HELLO messages received from other nodes. They assume that the processing required by a wormhole adds a delay function to the messages and argue that its effect can be used to determine whether the messages are subject to a wormhole attack. While their assumption should be valid concerning the processing required by a wormhole, they also assume that an attacker would randomly drop packets and appear to assume that without the presence of a wormhole the packet loss rate is practically zero. Both of the latter assumptions appear invalid and contribute to the distinguishability of the patterns observed for legitimate versus illegitimate links. We thus doubt that the approach would work under more realistic conditions.

Another statistical approach is suggested in [67]. They determine the frequency of links appearing in routes, assuming that outstanding frequencies should be caused by wormhole attacks. However, their performance analysis suggests that the rate of false alarms will quickly rise to unacceptable levels with an increasing network size. This comes at little surprise since a high link frequency usually occurs for any kind of bottleneck in a network.

In the same paper, a trust or reputation based approach is presented, based on the assumption that a wormhole would forward frames required for establishing routes but would drop a significant ratio of the other frames. This allows this and other trust or reputation based approaches (e.g. [53, 65, 70]) relying on this behaviour to monitor the retransmission of frames sent through links and rate their trust level based on the ratio of packets that were verifiably delivered through a given link, possibly sharing this information with other nodes. An attacker might however elect not to drop any or no significant fraction of the frames it captured or use the approach we suggested for similar schemes in Section 3.1, transmitting the frame to random layer 2 destination addresses to create the impression that the frame was retransmitted, rendering these approaches unable to detect the attack. We thus have to conclude that using the given methods, neither statistical nor reputation based systems appear to provide a reliable base for detecting a wormhole attack.

[9] presents a very recent topology based approach. It assumes that a central instance is able to obtain a connectivity graph representing the topology of the given network. It will then start at a chosen node and eliminate edges referring to that node and its immediate neighbours. The authors argue that the count of links traversed by a route from one node's

second degree neighbour to another should not change significantly by eliminating these edges, more particularly, there should always be another route that does not traverse more than twice the number of edges that a route required before eliminating the given edges. Correspondingly, the network is assumed to be subject to a wormhole attack if no such route exists. Their approach will however fails to distinguish a situation where two partitions of a network are connected through a single set of nodes which are within each others transmission range from a wormhole attack. It also requires wormhole links to decrease the hop count of a route between at least one set of nodes by more than half. In MANETs with tens of nodes, routes will generally traverse only few links and thus such an impact on connectivity may be impossible to obtain. Thus, we conclude that the approach may not be appropriate for MANETs.

[33] describes an approach that received widespread attention. They use directional antennas, transmitting a HELLO message using one antenna at a time, nodes receiving the message will respond, including the antenna through which they received the message. If the node received the HELLO through an antenna which was not opposing the sender's respective antenna, the sender will ignore its reply. While not stated explicitly, this assumes that all arrays of antennas will constantly face the same direction throughout the process and may not be tilted. If the HELLO was received through the expected, i.e. opposing, antenna, the replying neighbour will be added to a set of assured neighbourships and receive a notification by the sender. This indicates however that the protocol would provide a neighbourhood table that another protocol could use to route in the respective network. To ensure effective routes even in the face of node mobility, the described message exchange would have to be repeated periodically. The authors of [33] suggest using six directional antennas, requiring the sender of a HELLO message to transmit six messages for each update interval where each true neighbour will result in the exchange of another two messages. Given these figures, it appears unlikely that an acceptable trade-off between energy and bandwidth consumption and the ability to establish and maintain routes in a mobile network can be found.

The final paradigm we want to cover here are location based approaches. Most prominently, [34] introduced the concept of "geographical leashes", a mildly revised version of the same paper has been released as [35]. It assumes that nodes have loosely synchronised clocks and a means to determine their position, e.g. through a GPS, GLONASS or Galileo device attached to them, and include it along with the timestamp of transmission, both of them digitally signed, in each transmitted packet. A node receiving a packet may thus determine the distance a packet travelled according to its own and the position included by the sender. To counter a number of effects described below, the recipient will however compute an upper bound for the distance d_{sr} that the packet travelled, which will include not only the sender's and recipient's positions p_s and p_r but also the difference between the timestamps t_r determined by the recipient and t_s included by the sender, maximum speed of a node ν, possible synchronisation error of clocks Δ and relative position error δ:

$d_{sr} \leq |p_s - p_r| + \delta + 2\nu \cdot (t_r - t_s + \Delta)$

Reasons for including the position error are that current positioning systems are not able to provide an exact position but only an approximation thereof with accuracy reduced by effects such as reflections and refractions of signals caused by tall buildings or mountains. Also, each of the nodes may have moved after the position was written to the packet by the sender, thus the maximum distance they may have travelled depends on their speed and the difference between the timestamps t_s and t_r but also on the maximum clock drift between the two, denoted by Δ. An operator may now define a threshold for d_{sr}, depending on the maximum transmission range that he expects to achieve with his equipment, to classify each packet that travelled beyond that distance as a wormhole packet.

While the concept is sound and provides the base for the wormhole detection mechanism we describe in Section 4.3, the described mechanism has several shortcomings. First of all, it requires the authentication of a part of all packets received by each node. This is mitigated in part by employing the TIK protocol[1] also described in [34]. It reduces authentication to a couple of symmetric cryptography operations but requires that the authenticator received an authenticated key from the node generating the signature, which would typically imply the use of asymmetric cryptography. Also, the protocol requires that these keys would be distributed among the possible neighbours of each node in a network and will only provide for a limited number of authentications, i.e. this step may have to repeated periodically. In addition to that, choosing proper values for the maximum transmission range and speed of nodes may be difficult in heterogeneous networks. Setting them to the minimum or maximum values in such an environment may result in high rates of either false positives or negatives, particularly when nodes may differ significantly, e.g. when considering infantry soldiers and jet air planes. This matter is however not addressed in [34].

[62] describes several improvements to the OLSR protocol to improve its security. They use geographical leashes for protection against wormhole attacks but point out that location data may be unavailable in some situations. If a node is able to determine its own position but the neighbour to verify is not, it will determine the ratio of location aware neighbours of that neighbour that are at a distance to itself of at most twice the maximum transmission range and interpret this as the probability of the original neighbourship being genuine. For two nodes that are unable to determine their location, the protocol suggested in [62] checks whether the node's location aware neighbours are within at most three times the maximum transmission range and again interprets the respective ratio as the probability of the link being unaffected by a wormhole attack. Since this approach is mainly subject to the shortcomings we pointed out considering the geographical leashes concept, we abstain from discussing further details. A strikingly similar approach is presented in [39], however on the base that some nodes may not be equipped to determine their location. Since most recent cellphones are equipped with GPS chips, it appears unlikely that a MANET that should be protected from sophisticated attacks should contain any nodes without a means to determine their own position. Searching for means to replace this requirement thus appear out of place.

3.4. Summary

This chapter provided an overview to the literature concerning the detection of the sinkhole and wormhole attacks as well as link quality forgery. Approaches concerning both of the former attacks tend to make use of asymmetric cryptography which may however be too resource consuming in MANET environments. Several publications describe signatures for detecting sinkhole attacks which however appear to rely on an unrealistic model of the wireless channel, other approaches would severely affect the efficiency of the use of the wireless channel and are thus unfavourable in a resource-constrained environment such as a MANET. Finally, we were

[1]We do not provide a detailed discussion of this cryptographic protocol since that would go beyond the scope of this document.

able to suggest a scheme that would sidestep the main method for measuring trust in reputation based schemes, indicating that these approaches will not work against a sophisticated attacker.

Our next focus were schemes to detect forged link qualities. The approaches published so far were all tightly coupled with either the routing protocol employed, the sensor network environment they were developed for, or both. Their assumptions, such as the ability to determine maximum link qualities for given links, immobility of nodes or uniform transmission power levels, render them inapplicable in MANET environments.

Finally, we discussed approaches for detecting the wormhole attack in MANETs. Most timing based approaches fail to consider the delays introduced by a OSI layer 2 protocol such as IEEE 802.11. While one approach sidestepped this challenge by integrating the respective protocol within the second layer, it suffers from the aforementioned extensive need for asymmetric cryptography operations and would require hardware modifications. Other approaches were using the trust or reputation paradigm but assumed that an attacker could be identified by his unwillingness to forward packets. We found that this would not only contradict the nature of a wormhole attack but that these approaches would also fall for the scheme we suggested for sidestepping these methods when applied for detecting sinkhole attacks.

Besides a topology based approach, we also briefly described an approach using directional antennas which received significant attention in the research community but appears to be impractical due to the heavy load it would introduce on the wireless channel. The final paradigm for detecting wormhole attacks we discussed were location based approaches. While these would require extra hardware, namely a means to determine a node's position, we consider these approaches more practical than the previously discussed ones. Since the concept of geographical leashes is not only the most prominent among these approaches but also provides the base for the approach we will present in Section 4.3, we provided a more detailed discussion of its features, including those that require further elaboration, and introduced two recent approaches that were based on the concept.

Our literature research revealed a large volume of creative approaches to mitigate the attacks we want to consider in this thesis, far too many to present all of them in this chapter. Each approach however displayed some weaknesses, most of which we tried to point out in this chapter. We are thus confident that the approaches we use in this thesis provide a valuable contribution to the effort of detecting the aforementioned attacks in MANETs.

Chapter 4

The ToGBAD Detector for MANET Routing Attacks

The Topology Graph Based Anomaly Detector (ToGBAD) consists of a small set of detectors for routing attacks in MANETs. While the ideas underlying the individual detectors may differ significantly, they all revolve around the concept of some or all nodes acquiring data available or derivable locally and forwarding it to a dedicated node where it will be aggregated by a central detector to determine whether the network is currently being attacked. This structure is very beneficial in tactical MANETs where most nodes have tight constraints on size and weight, effectively limiting their capabilities in terms of processing and battery power. On the other hand, most of these networks are have some nodes which are less constrained, e.g. because they are mounted on a vehicle or associated with an object that has lower requirements in terms of mobility. Such a node would run the centralised components of our detectors, so that they are less restricted in terms of processing power required for their detection process without affecting the node's user experience.

In addition to the aforementioned benefits, centralised detectors provide additional advantages over purely distributed approaches. First, the availability of data from several nodes provides a broader basis for distinguishing attacks from normal behaviour. This may also render attacks against detectors less practical where nodes try to accuse other nodes of perpetrating an attack which are actually not doing so. Finally, to secure reports against manipulation or counterfeiting and to ensure their confidentiality, sender and receiver require either a common shared key or have to rely on asymmetric cryptography for encryption and authentication. While the former may require the distribution of a large number of preshared keys if communication is to take place between arbitrary nodes in a network, the latter is prohibitively expensive in terms of processing power. Since in our scheme it is predetermined that the recipient of the reports is the node running the centralised detectors, each node would require only a single shared key with that node in order to secure its reports using symmetric cryptography.

In the next section, we describe the ToGBAD detector for sinkhole attacks, followed by a description of our detector for forged link qualities. Last but not least, we introduce the ToGBAD detector for wormhole attacks in Section 4.3. This chapter closes with a brief summary in Section 4.4.

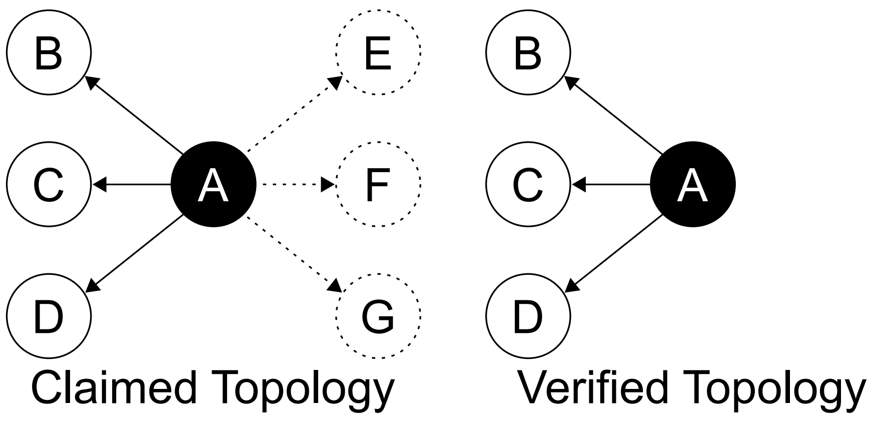


Figure 4.1.: The ToGBAD method: An attacker, node Ⓐ, claims, in addition to the existing neighbourships to nodes Ⓑ through Ⓓ, to have neighbourships with nodes Ⓔ through Ⓖ. Since only the existing neighbours report receiving HELLO messages from Ⓐ, the verified topology graph includes only three neighbours while Ⓐ claims to have six, revealing its malicious behaviour.

4.1. The Topology Graph Based Anomaly Detector for Sinkhole Attacks

The Topology Graph Based Anomaly Detector (ToGBAD) is the original ToGBAD method and is still included in the name of all the detectors that evolved around its basic principles. While we describe it briefly here, we refer to [27] for further details regarding this approach. ToGBAD is a detector for sinkhole attacks in MANETs using a proactive routing protocol with neighbourship announcements, such as Neighborhood Discovery Protocol (NHDP) or OLSR. All nodes in the network send periodic reports to the central detector, indicating from which nodes they received locally broadcast routing messages since the last report was sent and how many neighbourships were announced in those messages. Since such messages are usually termed HELLO messages, we will refer to them by that name below.

These reports first of all allow the central detector to build a graph of mutually confirmed neighbourships, the ToGBAD graph. The graph will contain an edge between two nodes if and only if both of the nodes sent a report indicating that they received a HELLO message from the other node and a predefined timespan ρ has not been exceeded since receiving those reports. This graph may now be used as indicated in figure 4.1. Since it will only contain neighbourships confirmed by both nodes, an attacker will only have confirmed neighbourships between itself and its real neighbours and possibly colluding attackers. More particularly, claimed but non-existing neighbourships to legitimate nodes generally do not exist in the ToGBAD graph. Thus, the detector may simply compare the reported count of neighbourships a node is advertising against the count of verified neighbours. If the former is larger than the latter, this indicates that the node may be executing a sinkhole attack.

While the method we described in the previous paragraph provides a good background for detecting a sinkhole attack, it may lead to ToGBAD falsely detecting an attack if reports arrive late or get lost completely. This is countered by several mechanisms. First of all, ToGBAD starts with a configurable length learning phase during which no detection takes place, giving the graph some time to be populated with neighbourship information. After the learning phase, ToGBAD will start monitoring the difference between the count of neighbours confirmed through the previously described method and the values reported by the nodes. If the difference is not considered to be suspicious, it is incorporated in an aged mean and aged deviation value of the differences, providing an indication of the overall accuracy of the ToGBAD graph for non-attacking nodes. Thus, a difference is considered to be malicious if and only if it exceeds the aged mean value by more than the aged deviation multiplied by a user defined factor. Since forging a single neighbourship provides little benefit to an attacker, a difference value of 1 will be tolerated regardless of the aged mean and deviation values.

More formally, if a subject was reported to have claimed neighbourships to ϕ nodes while the ToGBAD graph contained ψ verified neighbourships, we define the difference as $\delta = \max(0, \phi - \psi)$. We do not permit the difference to drop below zero since this value would typically not occur for an attacker while allowing negative values might reduce the tolerance towards inaccuracies in the ToGBAD graph induced by using this approach too quickly or even lead to negative tolerance through the mechanism described below.

We define the aged mean μ_i and aged deviation σ_i to be 0 for $i = 0$. To calculate μ_i for $i > 0$, based on the ith observation δ_i and previous mean μ_{i-1}, we also consider a user defined weight for new observations $\alpha \in [0, 1]$:

$$\mu_i = \mu_{i-1} + \alpha \cdot (\delta_i - \mu_{i-1})$$

The aged deviation σ_i considers, in addition to δ_i, σ_{i-1} and μ_{i-1}, another user defined weight factor, $\beta \in [0, 1]$:

$$\sigma_i = \sigma_{i-1} + \beta \cdot (|(\delta_i - \mu_{i-1})| - \sigma_{i-1})$$

Note that [23] and [29] provided the same definition for σ_i, but conflicting definitions for μ_i. Their definitions are however equivalent in that by defining $\alpha_{\text{new}} = 1 - \alpha_{\text{old}}$ one can be translated into the other. In this thesis, we use the definition provided in [23] so that both α and β refer to the weight of the most recent observation.

To determine whether an incoming value δ_{i+1} indicates an attack, we check whether the following assumption holds:

$$\delta_{i+1} > \max(1, \mu_i + \omega \cdot \sigma_i)$$

where $\omega \in \mathbb{R}^+$ refers to another user defined factor, indicating by which magnitude of σ_i δ_{i+1} may exceed μ_i. Since by the definition of the two variables, values of up to $\mu_i + \sigma_i$ are to be expected by non-attacking nodes, we restrict ω to $\mathbb{R}^+ \setminus [0, 1[$.

4.2. ToGBAD Link Quality

When a routing protocol is used that supports link qualities, an attacker perpetrating a sinkhole attack can use this to leverage its attack by also forging link qualities. ToGBAD Link Quality (ToGBAD-LQ) describes a method to prevent this kind of attack. As for the method described in the previous section, it requires that the routing protocol uses periodically and locally broadcasted messages and link qualities are determined based on their reception, which is a common method when there is no a priori knowledge on the sender's transmission power. In this section, we will again call the respective messages HELLO messages without making further assumptions about the employed routing protocol other than the aforementioned ones.

Determining link qualities in the fashion described in the previous paragraph yields a quality estimate for a given unidirectional link but does not necessarily reveal any information about the link in the reverse direction. Link quality metrics such as the ETX metric briefly described in Section 2.2 thus rely on nodes sharing these link qualities so that each node can combine the values determined by two adjacent nodes to obtain an estimate for the bidirectional link quality for a given link.

This property of determining link qualities leaves a path open for attacks. An attacker would usually repeat the link qualities reported for the unidirectional link from itself to its neighbour, the neighbour link quality (NLQ), since checking their correctness is trivial. It could however forge the quality for the reverse link since a node usually has no way to verify the claim for that link. At this point, the local part of ToGBAD-LQ comes into play, employing two mechanisms. First, it will store all NLQ values transmitted by the node and within its own link quality window. Thus, it may verify whether a node forges NLQs concerning itself. The second mechanism uses a challenge/response based method for authenticating link quality claims for the reverse direction. Nodes detecting inconsistencies through either mechanism will inform the central detector which will determine whether a node appears to actually be perpetrating an attack.

Consider a node Ⓐ sending a HELLO message as a part of its neighbourhood detection process. ToGBAD-LQ will include a challenge in that message, so that any node Ⓑ receiving it may generate a response based on both the challenge and its own ID. When Ⓑ generates its own next HELLO message, it will include that response in the message. Node Ⓐ will, upon receiving the response, verify its correctness and store the result of the check in a local database. It may then resort to the stored information on valid responses received from Ⓑ to verify whether its claim for the quality of the link from Ⓐ to Ⓑ appears legitimate.

Since HELLO messages, including the aforementioned responses, are subject to the packet loss on the given unidirectional link, the process of verifying these claims has to consider the quality of that link. For a claim of link qualities α and β, each with values ranging between and including 0 and 1, for the links between a verifying node and a neighbour, the verifier will thus calculate the ratio γ of responses that should have been answered correctly simply by multiplying the two:

$$\gamma = \alpha \cdot \beta$$

Since simply comparing the ratio of correctly received responses against γ is likely to cause false alarms for small but non-malicious deviations that occur e.g. when a HELLO containing more than one response is lost, we add a tolerance τ of responses assumed to have arrived in addition to the confirmed ones. When λ is the number of correct responses received and ω the window size, we calculate the verified response ratio as:

$$\nu = \min(1, \frac{\lambda + \tau}{\omega})$$

Finally, we determine the difference $\delta = \gamma - \nu$ between the two. If it is positive, the claimed link quality exceeds the value verified by previously received responses, indicating that an attack is taking place. Note that since we add τ to the count of correct responses, the minimum value for ν is $\frac{\tau}{\omega}$, implying that δ will never exceed $\frac{\omega-\tau}{\omega}$. To render our method more robust against small deviations in local configurations and be able to provide values for thresholds which are not tightly coupled to a given configuration, we normalise δ by multiplying by the inverse of this maximum and returning 0 instead of negative values:

$$\delta_{\text{normalised}} = \max(0, (\gamma - \nu) \cdot \frac{\omega}{\omega - \tau})$$

Note that this requires that $\tau \neq \omega$; setting τ to ω would however imply that any claim would be considered correct and thus disable the detection mechanism altogether. We may thus assume that this requirement is met by any sane configuration.

As mentioned previously, upon encountering a value that appears to be malicious a node will report the given value to the central detection instance of ToGBAD-LQ where reports from several nodes will be aggregated to provide a more robust classification of possible attacks. The first implementation of ToGBAD-LQ was described in [10] and aggregated messages by calculating the mean value of all reports received for a given node and within a user defined timespan ρ. A reimplementation based on that concept and described in [12] indicated that using another method for aggregating reports might yield better trade-offs between false positives and false negatives. We thus provide not only a more formal description of the original method in Section 4.2.1 but also present two alternative methods in the sections following thereafter. These approaches will be evaluated in the course of this thesis.

4.2.1. Aggregation Method: Mean

This was introduced as the first method for aggregating reports received by ToGBAD-LQ's central detector. It calculates the mean value of all reports referring to a given possible attacker and received within a predefined timespan ρ. Since small deviations, mostly caused by lost or delayed responses, are common observations but large reported values are likely to refer to an attack, the mean value reflects a tendency for the reported values which will be compared against a user defined threshold χ to distinguish between an attack and non-malicious behaviour. A second user defined threshold π protects the detector against attacks where an attacker tries to reduce the mean value obtained by sending a large number of messages with small values. Messages reporting values below π will simply be ignored. The pseudo code below describes the process of determining the mean value:

```
X = {$x_i$ | $x_i$ is a link quality report received by the detector}
a(x) $\mapsto$ attacker reported in x
r(x) $\mapsto$ node that sent report x
t(x) $\mapsto$ time of receiving x
$\delta$(x) $\mapsto$ value reported in x
m # most recently received report
c = 0 # count values reported concerning a given node
s = 0 # sum of reported values concerning a given node

for each x $\in$ X:
  if a(x) = a(m) and t(x) + $\rho$ $\geq$ t(m) and $\delta$(x) $\geq$ $\pi$
    s = s + $\delta$(x)
    c = c + 1
  endif
endfor

if c > 0
```

```
  mean = $\frac{s}{c}$
else
  mean = 0
endif
```

Finally, the mean value determined above will be compared against the threshold χ and an alarm will be generated if and only if `mean` $\geq \chi$.

4.2.2. Aggregation Method: Mean with Node Threshold

A node will usually not forge link qualities with respect to a single neighbour but for a set of nodes including some or all of its neighbours and possibly even nodes that are not among its neighbours at all. Thus, if only a small number of nodes report forged link qualities, these reports are unlikely to have been caused by an attacker, since forging link qualities for a few nodes only yields little benefit to the attacker. This approach thus starts out by determining the set of reports to consider for aggregation R almost exactly as in [12], however we also maintain a set of nodes N that contributed reports to R, allowing us to compare the cardinality of N against a user defined threshold η. Note that in the pseudo-code below, ρ refers to a user defined time period to consider for aggregation and π is a user defined “lower threshold” that should prevent colluding attackers from lowering the mean value regarding their peer by sending a large number of reports with low values:

```
X = {x$_i$ | x$_i$ is a link quality report received by the detector}
R = ∅ # chosen reports
N = ∅ # reporters for chosen reports
a(x) ↦ attacker reported in x
r(x) ↦ node that sent report x
t(x) ↦ time of receiving x
$\delta$(x) ↦ value reported in x
m # most recently received report

for each x ∈ X:
  if a(x) = a(m) and t(x) + $\rho$ ≥ t(m) and $\delta$(x) ≥ $\pi$
    R = R ∪ x
    N = N ∪ r(x)
  endif
endfor
```

After executing this algorithm, we compare the count of elements $|N|$ against η, if $|N|$ is below η, we assume that too few nodes contributed to R for R to be revealing an attack. Thus, if and only if, $|N| \geq \eta$ we compare the mean Δ of all reports in R against the aggregation threshold χ:

$$\Delta = \frac{\sum_{r \in R} \delta(r)}{|R|} \geq \chi$$

4.2.3. Aggregation Method: Node Picking

Our final approach determines the number of nodes that appear to be under attack by a given node. We assume that effects leading to exceedance reports even when no attack is taking place are distributed roughly uniformly at random but may result in a single node generating several reports regarding another node until the effect wears off. Thus, while reports regarding an innocent node may arrive at any time, several nodes reporting on an innocent node should be a rare occurrence.

This would indicate that using the approach described in the previous section with a reasonably high threshold η for reporting nodes should provide reliable protection against false alarms. However, an attacker could forge link qualities only for such nodes that appear not to be in its transmission range and would therefore not be able to detect its attack. Only nodes that were assumed not to be within the attacker's transmission range but either are in range or move into it, and thus receive messages with forged link qualities concerning themselves, would be able to report on such an attacker, providing an argument for a lower threshold η when using that method.

The goal of this approach is thus to allow reasonably low thresholds on the number of reporting nodes without raising the ratio of false positives to unacceptable levels. As indicated in the introductory paragraph, we do not aggregate the reported values as in the previous approaches but simply classify each individual reporting node as whether or not it is being attacked. This allows us to include reports on forged NLQ values in our decision making process. Note that since detecting a forged NLQ is a binary decision, there is no obvious way to include this information in a numeric aggregation.

Classifying nodes reporting forged link qualities is simple; whenever we receive a report regarding a given node, we check all reports regarding that node that arrived within a user defined timespan ρ. Each node that sent a report including a value of at least π is classified as "under attack", where π should be a small value or even zero, indicating that each node sending a report is considered to be under attack. Note that with this method, reporting forged values or having a colluding node report forged values no longer yields a benefit to an attacker since it would no longer influence the decision regarding other nodes. In a second step, we consider all nodes that detected forged NLQ values to be under attack. Finally, we consider the sum of the nodes that were classified as being under attack in each of these steps, eventually counting nodes that reported both forged link qualities and NLQs twice. We then compare this sum against a user defined threshold η and generate an alarm if it is equal to or larger than η. On a related note, when choosing $\eta \geq 3$, no single node can trigger an alarm for another node, limiting an attacker's ability to gain an advantage through falsely accusing other nodes.

The following pseudo code formally describes the presented algorithm:

```
X = {$x_i$ | $x_i$ is a link quality report received by the detector}
Y = {$y_i$ | $y_i$ is a neighbour link quality report received by the detector}
L = $\varnothing$ # nodes that detected an LQ attack
N = $\varnothing$ # nodes that detected an NLQ attack
a(x) $\mapsto$ attacker reported in x
r(x) $\mapsto$ node that sent report x
t(x) $\mapsto$ time of receiving x
$\delta$(x) $\mapsto$ value reported in x
m # most recently received report

for each x $\in$ X:
  if a(x) = a(m) and t(x) + $\rho$ $\geq$ t(m) and $\delta$(x) $\geq$ $\pi$
    L = L $\cup$ r(x)
  endif
endfor
```

```
for each y ∈ Y:
  if a(y) = a(m) and t(y) + $\rho$ ≥ t(m)
    N = N ∪ r(x)
  endif
endfor
```

When the algorithm determined L and N, we may simply compare the sum of their members against the user defined threshold η and generate an alarm if and only if it was exceeded:

$$|L| + |N| \geq \eta$$

4.3. ToGBAD Wormhole

The final ToGBAD approach is ToGBAD Wormole (ToGBAD-WH), a refinement of the geographical leashes concept introduced in [34] (cf. Section 3.3). It was developed as a part of [11] and improved in [41]; a full presentation of the approach can be found in [28]. The basic idea is, as with packet leashes, to obtain information about the distance frames have travelled and then determine whether a given frame travelled beyond the maximum transmission range of the sender. Since that could also be caused by small scale effects concerning the wireless channel, ToGBAD-WH allows considering not only single conspicuous links, as in the original packet leashes concept, but aggregates the respective information, triggering an alarm if and only if a given dynamic threshold of conspicuous links has been detected.

To discover conspicuous links, ToGBAD-WH requires the availability of both node locations as well as information on the connectivity in a monitored MANET at a central detector instance. While these requirements may appear expensive at first sight, nodes in a tactical MANET will often be equipped both with a means to determine their own position and to share this information with a node operated by a superior who would use the node's locations to direct the actions of their operators. If this node also runs ToGBAD-WH, there is no cost associated with determining the node's locations. Secondly, the ToGBAD approach requires that nodes share local connectivity with a centralised detector, so if the given superior's node runs not only the central detector for ToGBAD-WH but also for ToGBAD, there is no need to transmit any extra information.

As in [34], ToGBAD-WH considers the possibility that positions are outdated or inaccurate by introducing two methods for determining the distance between two nodes. $d_{\text{simple}}(A, B)$ refers to the obvious euclidean distance between two nodes A and B, disregarding the time that has passed since the positions were determined or possible inaccuracies of the positioning system. To take the latter into account in the detection process, ToGBAD-WH allows using the $d_{\text{advanced}}(A, B)$ method for determining the distance between nodes. It will consider the time that passed since the last position updates, denoted by δ_X for a node X, maximum speed of the nodes, denoted by s_X, and the maximum position error Δ that has to be expected with the system employed:

$d_{\text{advanced}}(A, B) = \max(0, d_{\text{simple}}(A, B) - \delta_A \cdot s_A - \delta_B \cdot s_B - 2 \cdot \Delta)$

Note that while this formula is close to the one presented in [34], there is a very important difference. The authors of [34] add the aggregated position error to their advanced distance, obtaining a maximum distance that a frame may have travelled, while we subtract it, assuming that the nodes may have moved towards each other between their last position updates. This allows us to use the expected maximum transmission range of a node as a threshold while their method would require multiplying the maximum transmission range by a tolerance factor to avoid triggering alarms when the distance between nodes increases but remains below the expected transmission range. On the other hand, this variation of the approach may yield a distance below 0, if nodes are within each other's proximity. Since this would contradict the definition of the term distance, we define $d_{\text{advanced}}(A, B)$ to be 0 in these cases.

To determine whether a given frame sent by a node A and received by a node B travelled a conspicuous distance, we simply compare the distance obtained by using one of the methods described above against A's transmission range or t_A:

$$t_A < d_{\{\text{simple}|\text{advanced}\}}(A, B)$$

The evaluation of a ToGBAD-WH implementation in [11] showed that solely evaluating the distance given frames have travelled yields a significant rate of false positives, thus ToGBAD-WH considers the monitored network to be subject to a wormhole attack if and only if frames between several pairs of nodes travelled beyond the sender's maximum transmission range in a user defined interval ρ. Determining a threshold for the count of conspicuous links required for triggering an attack is however non-trivial and was thus warranted further investigation in [41]. We summarise the algorithm developed in this work below.

The method introduced in [41] exploits the topology graph provided by the ToGBAD approach. For each connectivity report that indicates a frame transmitted by node A and received by another node B travelled beyond A's maximum transmission range, the algorithm determines the set β of verified neighbours of B through the ToGBAD graph. It will then however eliminate such nodes from β that would not be within B's transmission range and should thus not have been able to receive frames transmitted by B in the first place, yielding a subset $\beta' \subseteq \beta$ containing the "true neighbours" of B. Nodes in β' are supposed to be true in the sense that their neighbourship relation to B has not been caused by a wormhole attack.

If B was right next to the closest wormhole endpoint and that endpoint had a similar set-up in terms of the device used to retransmit captured frames, one would expect that each node in β' would be in the wormhole's transmission range, i.e. the central detector would receive a report indicating a frame that travelled a conspicuous distance from each node in β'. However, this assumption will generally not hold. Thus, $|\beta'|$ provides the basis for determining an automatic threshold but is multiplied by a user defined factor κ, where κ should be between 0 and 1, i.e. $0 < \kappa \cdot |\beta'| \leq |\beta'|$. Finally, we want to ensure that the algorithm does not permit single links to trigger an alarm, which would be the case for $\kappa \cdot |\beta'| < 2$, nor should it set the threshold too high in dense networks. Thus, the final threshold χ will also depend on another user defined upper threshold η. Putting these requirements together results in the following equation for χ:

$$\chi = \max(2, \min(\eta, \lfloor \kappa \cdot |\beta'| \rfloor))$$

With χ determined, ToGBAD-WH will determine the distance travelled by each frame transmitted by A and reported to have been received within a user defined interval ρ. Nodes for which a conspicuous frame has been discovered will be added to a set α, if the count of nodes in α is equal or larger than χ, we assume that the network is subject to a wormhole attack. The pseudo code provided below provides a more formal description of this algorithm:

```
X = {$x_i$ | $x_i$ is a connectivity report received by the detector}
A = ∅ # nodes reporting frames travelling conspicuous distances
$N_x$ = {$n_i$ | $n_i$ is a verified neighbour of node x in the ToGBAD graph}
$T_x$ ↦ maximum transmission range of node x
d(x,y) ↦ distance between two nodes x, y, may be 'simple' or 'advanced'
s(x) ↦ sender of the frame reported in x
r(x) ↦ node that sent report x
t(x) ↦ time of receiving x
m # most recently received report

# Step 1: Check whether the distance between the
# sender and reporter is conspicuous, abort, if not
if d(r(m), s(m)) ≤ $T_{s(m)}$
  return 'Not an attack'
endif

# Step 2: Determine the threshold $\chi$ based on the
# neighbourships of the node receiving the last report
$\chi$ = 0
for each n ∈ $N_{r(m)}$:
  if d(r(m), n) ≤ $T_{r(m)}$
    $\chi$ = $\chi$ + 1
  endif
endfor
$\chi$ = max(2, min($\eta$, floor($\kappa \cdot \chi$)))

# Step 3: Check how many nodes appear to have a
# conspicuous link to the sender of the reported frame
for each x ∈ X:
  if s(x) = s(m) and t(x) + $\rho$ ≥ t(m) and d(r(x), s(x)) > $T_{s(m)}$
    A = A ∪ r(x)
  endif
endfor

# Step 4: Compare the cardinality of A against
# the threshold $\chi$ determined above
if |A| < $\chi$
  return 'Not an attack'
else
  return 'Wormhole attack detected'
endif
```

4.4. Summary

In this chapter we introduced the Topology Graph Based Anomaly Detector (ToGBAD) detectors for routing attacks in MANETs. There are three approaches, the ToGBAD approach for detecting sinkhole attacks, ToGBAD-LQ for revealing link quality forgery and ToGBAD-WH for detecting wormholes.

ToGBAD builds a topology graph based on nodes mutually reporting the reception of locally broadcast frames. A node's claimed count of neighbourships will then be compared against the sum of the node's verified neighbourships in the topology graph and an aged deviation value reflecting the overall accuracy of these neighbourships for non-malicious reports. Thus, a node reporting a significant number of non-existing links should trigger an alarm.

The second approach we described in this chapter was ToGBAD-LQ, introducing challenges and responses in locally broadcasted routing messages to allow the verification of claims for link qualities determined by a given node's neighbour. While this part of the verification is carried out by each individual node, i.e. locally, a result indicating an attack is reported to a central detector for aggregation. The previously described aggregation method had not performed well in earlier research, so we described two alternatives that will be evaluated in the course of this thesis.

Finally, we described the ToGBAD-WH approach for wormhole attacks. It relies on connectivity and position reports to determine whether nodes appear to have received frames that travelled beyond the sender's maximum transmission range. With a recent addition, ToGBAD-WH determines dynamically how many nodes should appear to have received frames through such a conspicuous link before an alarm is being triggered. We expect that this significantly reduces the number of false positives compared with other position-based approaches, e.g. the packet leashes concept which we consider this approach to be a refinement of.

Chapter 5

Evaluation Environment

In this thesis we introduce the sinkhole and wormhole attacks with the goal of assessing their impact. We also introduce the ToGBAD intrusion detection mechanisms. In order to determine their effectiveness in detecting the aforementioned attacks, we need to construct an environment that allows us to carry out these evaluations. Choices are generally simulations, emulations and field tests where cost and complexity generally increase in that order. Since our implementations use real software which cannot be interfaced with simulations, our choice was limited to the latter two options; field tests were ruled out by the cost and challenges in being able to provide reproducibility among several tests.

We introduce our emulation environment in Section 5.1, which also provides an overview to the challenges we faced when setting up our emulation environment. The following Section 5.2 gives an overview of the scenarios used for our evaluations and finally in Section 5.3 we describe the measures we took to achieve an acceptable level of reproducibility given the aforementioned challenges. The chapter closes with a brief summary in Section 5.4.

5.1. MANET Emulation Environment at University of Bonn

Emulation is a method which is based on the principle of using real components and combining them with virtual components to obtain results that are more accurate than simulations but at lower complexity and cost than incurred by a full deployment, particularly when testing prototype equipment and implementations. As an additional advantage when compared with deployments or other means of field testing, the reduced complexity may make it easier to reproduce environment conditions, allowing the evaluation of different components or configurations under otherwise exactly the same conditions.

At the University of Bonn, Dept. of Computer Science 4, emulations are a means of testing and evaluating software developed for MANETs. The set-up used for this thesis consisted of two computers equipped with a dual core CPU, 4GBs of RAM and a 250GB SATA hard drive each. During the course of this thesis, we added another computer with a similar configuration to resolve issues described in Section 5.3. They were connected to the Internet through the regular university network and a second private network link. We set them up with Debian Linux, running a custom kernel that includes the OpenVZ extensions. The latter allows running several containers or virtual environments (VEs) on each host without providing a virtual machine or a separate kernel for each container. OpenVZ ensures that the containers will neither be aware of each other nor be able to access any memory or file system out of their own scope. These containers will act as wireless clients in our emulation set-up.

While each VE is set up with a virtual Ethernet interface that we connected to a Linux kernel bridge to provide the containers with network connectivity, we use a second piece of software, the MotionEmulator^{NG} (ME) to provide virtual wireless NICs. After a brief overview of the node's set-up in Section 5.1.1, the section thereafter gives an introduction to the ME software and describes most of the challenges we faced using it and our approaches to solving them. Finally, we give a short introduction to the Diminutive Breadroll Emulation System (DBrES), our system for preparing and controlling emulations in Section 5.1.3.

5.1.1. Node Set-up

In our emulations, each node is represented by a VE, connected to the emulation network as described in this section's introduction. They use minimal configurations of the Debian Linux distribution but we also installed the Responsive Intrusion detection for Tactical Ad hoc networks (RITA) project's software for lightweight nodes in all of them and the software for a fully equipped node in one of the environments used for this thesis. Since at least some of them were not relevant for our evaluations, we disabled most of the RITA components, leaving only the NetControl component for starting and configuring the RITA software and set it up to start the ME client, MessageEngine, KeyManager, Heartbeat and Flooding components.

To provide routing in our network, we use the OLSRd. While it is also used in the RITA project and can be set up through NetControl, we decided to start and configure it through a DBrES component, which provided more flexibility when working with the custom set-up required for our evaluations.

5.1.2. RITA Motion Emulator^{NG}

The MotionEmulator^{NG} (ME) was developed in order to provide an emulator for virtual MANETs for the RITA project and consists of a client and a server component, the MotionEmulator^{NG} Server (MES). Each client connects a virtual or hardware node with an instance of the MES and provides a virtual interface to the node. Whenever a frame is sent through the virtual interface, the ME client forwards it to the MES it is connected to, which will then decide which nodes should be able to receive that frame and forward it to the appropriate clients.

To model the wireless channel and its properties, the MES has to be set up with a scenario file, indicating the assumed positions and changes thereof for a set of nodes, and a configuration for an "oracle". Each oracle provides a different method for determining the reception of frames sent through the modelled wireless channel. The "hub" assumes that each node will receive any frame transmitted by any other node while the "TwoRayGround" oracle is set up with a maximum transmission range and each node within the given distance from the sending node will receive a given frame. The "Fading" oracle replicates the behaviour of the predecessor of the ME and uses a fixed, distance-based probability curve, i.e. the probability of a node receiving a frame is directly correlated and only correlated to the distance between that node and the sending node. Finally, the "RiceanFading" oracle provides a reimplementation of the Generic Fading module developed by the University of Bonn, Dept. of Computer Science 4

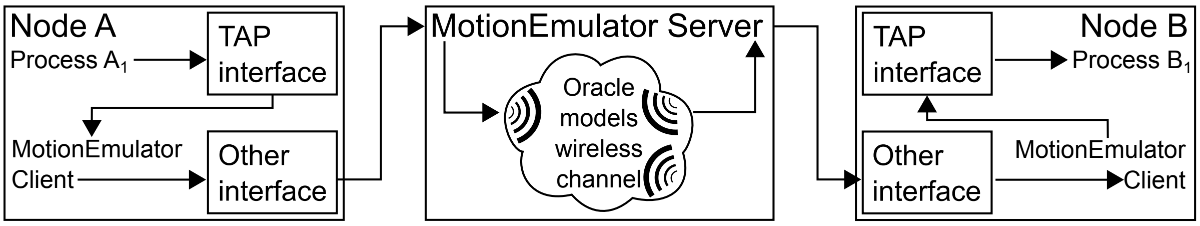


Figure 5.1.: The path of a transmission through a ME set-up. A process A_1 in node A generates a packet which will be sent through the TAP interface provided by the ME Client. The client uses a TCP connection to relay the frame generated by the TAP interface to the ME Server. An oracle module of the MES models the wireless channel, i.e. decides which nodes should receive the given frame. Once receptions have been determined, the frame will be forwarded to the ME Client on the respective nodes, in our example node B, which will inject it to its TAP interface. At that point, the node's kernel decides whether or not to discard the frame or, as in the depicted scenario, relay it to a userspace programme.

for the ns-2 Network Simulator (NS2) [21]. The module determines the signal strength for each receiver of a frame by taking into account both a dominant distance-based component and a randomised small-scale-fading component. It then checks the resulting signal strength against the assumed receive threshold of the modelled wireless NIC to determine whether a node would receive a given frame. The path of a packet travelling through an emulated network based on the ME is sketched in figure 5.1.

While the original MES would neither consider latency nor channel occupancy or collisions, it was extended as a part of [32] to include these properties of a wireless network. First, we give a brief summary of the reasons why we wanted to use this extended version of the MES in the next section. The sections following thereafter provide an overview of the mechanism introduced by it and finally the changes that turned out to be necessary to allow us to deploy it in our environment.

Reasons for Switching to the Extended Motion Emulator

In the course of this project, we ran emulations using the implementation of the wormhole attack described in Section 6.2.2. As expected, this would significantly increase the rate at which frames would be sent through the MES. Apparently, this also increased the severity of a problem whose effects we had already observed and described in [12], leading to clients losing their connection to the MES and rendering the results obtained irreproducible. While we were able to detect the condition and mitigate it by restarting affected emulations, almost no emulation would ever turn out to be affected to an acceptable degree when the wormhole attack was being executed within them. Since there was an obvious link between the load on the MES and the ratio of successful emulations, we were confident that the extended MES, which introduces delays to improve the realism of the modelling of the wireless channel, would mitigate these problems. We would like to add that the extended version did not work when we tried to integrate it into our environment during the project described in [12] and that we had to invest significant effort in the course of this project to ensure that it would. These efforts are described in Section 5.1.2 below.

Collision Extension

As a part of [32], the MES was extended to consider collisions, channel occupancy and their handling in IEEE 802.11 wireless networks. Each client connected to an instance of the MES is handled by its own instance of the ritaMotionClientHandler class which also stores its minimum back off or the timestamp of the earliest transmission possible by the associated client and a list of frames received by the client that have not yet been transmitted. A single instance of the ritaMotionCollision class models the interaction of the wireless NICs and the wireless channel. It polls the client queues to add up to one frame for each client to its own queue, the "main queue". Frames in the main queue will be considered for transmission once the sending client's back off has expired, the Collision Extension will however check whether another transmission is overlapping with the scheduled transmission and, if that is the case, whether the signal level of the earlier transmission is either above the receive or carrier sense threshold of the node that scheduled the later transmission. If so, the node's back off will be increased and the later transmission will be delayed. Otherwise, a collision occurs and the ability of clients to receive the later transmission will depend on the ratio by which the power levels of the two overlapping signals differ. To allow this mechanism to work, frames that have been transmitted but may still result in a collision will be marked but not removed from the main queue. They will however not be considered when polling clients that have no frame in the main queue for new frames.

With the introduction of the mechanism described in the previous paragraph, the modelling of retransmissions was also subject to significant changes. While the previous implementation made several calls to the oracle modelling the wireless channel if the unicast destination would not receive the given frame, the extended version will place a copy of the frame not received by its destination in its main queue and update the client's back off. The frame will thus be processed again in the manner described in the previous paragraph until it is either received or it hits the threshold for retransmissions.

Improvements and Bug Fixes

This section describes most of the changes that were necessary to the extended version of the MES before we were able to deploy it in our environment. While most changes were fixes to bugs which we encountered when trying to set up our environment, we also added some features and which we will also describe below.

Consistent Modelling of Time and Queue Deadlocks The original implementation of the MES used signed integers to model the emulation time in milliseconds that passed since the assumed start of an emulation. This had the benefit of allowing the definition of negative start offsets, if an emulation was started with a negative offset, it would remain in its initial state until the absolute value of that offset had passed. This made it possible to trigger initialisations etc. without affecting the actual emulation time. The resolution provided by these milliseconds (10^{-3}) timestamp was however too low to model the details of the wireless channel provided by the extension described in the previous section. Thus, the author of [32] used long unsigned integers to model offsets in microseconds (10^{-6}) in his extension, but left the modelling of time unchanged elsewhere.

This unfortunately caused a deadlock on both the main and client queues when using at least one of two settings provided by the MES. The first was the use of negative start offsets described above; since integer arithmetic for unsigned integers implies that -1 is the largest number, frames enqueued before the start of the emulation would not be transmitted after the emulation offset rolled over to positive numbers. Also, queue elements were dequeued in the order of arrival, i.e. any queue would be in a deadlock if it contained a frame that had been received at a negative time offset. While a client queue could be released from this deadlock by frames being dequeued and inserted into the main queue, this would inevitably cause a deadlock on the main queue which would then spread to all client queues since their queues would never be emptied. Another way to trigger this deadlock would be to use a feature of the MES that would repeat a scenario once its configured duration had passed. In that case, again timestamps that were lower than timestamps recorded for the scheduled transmission of previously received packets would be encountered and thus the same effects as described above would occur.

While a quick solution could have been to adapt the Collision Extension to use signed long integers for storing timestamps, we were uneasy about the use of different models for time in a single programme in the first place. Thus, we decided to replace any reference to timestamps with our SignedTimeval class.[1] The second problem described in the previous paragraph could however not be solved by this since it is caused by the non-linearity, not the representation, of time during the emulation. To address this problem, we introduced wall clock timestamps for all matters related to the client and main queues.

Non-Linearities Affecting Radio Propagation Model Solving the inconsistencies described above did not, however, solve another problem concerning timestamps that would occur when using the RiceanFading oracle with its coherence setting enabled. The Collision Extension would occasionally issue calls to the oracle to determine the signal strength at a receiver when a collision occurred. To determine the influence of the previously determined signal strength, a formula is evaluated which depends on the timestamp of the current call but also that of the last time the signal strength was determined between the respective nodes. However, when doing so in the context of collisions, the "current" timestamp would sometimes turn out to be earlier than the timestamp stored by the oracle. We were surprised to learn that the MES would not crash when this occurred because it would imply that a root would be extracted from a negative, non-complex number. However we found out that this would attenuate signals over time, ultimately resulting in nodes being isolated because their transmissions would always result in signal levels that were below the other node's receive thresholds. Since finding a correct solution for this issue would be a rather complex task and disabling the RiceanFading's coherence setting would not severely affect the realism of the wireless channel's model, we decided to simply disable it in our evaluations.

Segmentation Faults Two further issues we discovered resulted in frequent segmentation faults. The first one was that the reception and receiver signal strength determined for a frame would be stored in arrays with a fixed size corresponding to the maximum number of clients

[1]The class SignedTimeval represents timestamps using the types defined by the operating system for storing microsecond precision timestamps as a touple of seconds and microseconds in a struct timeval. Other than the struct timeval standardised in POSIX (cf. [3]), it comes with a boolean to indicate whether or not these values refer to a positive or negative number, i.e. it can represent both positive and negative values at microsecond precision. To ensure that code using that class remains easy to read, we used the feature of the C++ programming language allowing us to define operators on arbitrary types to define both operations and comparisons on instances of SignedTimeval and some related types.

the server is able to handle, which would be determined at compile time. To read or write the respective values for a client, its ID would be used as an offset to these arrays. The ID however is assigned through a registration message sent by an ME client after establishing a connection to the MES. This yielded two problems: first, an attacker could wilfully choose an ID that would point past the boundaries of these arrays to execute a denial of service attack against the MES. Since we were operating in a secure environment, the second issue was however more relevant to us. To determine the correct ID of a client, an exchange of messages between the MES and the respective client is necessary, thus the MES will assign temporary IDs to each client that would be overwritten once the registration message was received. These temporary IDs will be the count of connected clients plus the maximum number of clients the server may handle. Obviously, using this as an offset to an array of size "maximum number of clients" results in a segmentation fault. To overcome this without excessive changes to the internal workings of the MES, we simply added range checks whenever the ID is used as an array offset.

The second cause for segmentation faults was much harder to uncover. Each frame is stored along with a pointer to the client handler objects for the source and destination of that frame. For broadcast frames, the destination pointer is set to `NULL`. Whenever a client is disconnected from the server, the respective client handler and thus any pointer to that handler in the queues becomes invalid; hence any queued frame referring to it should be removed. While this was properly implemented for regular packets, the implementation skipped the check whenever the destination pointed to `NULL`, eventually leaving frames in the queue that referred to an invalid source and caused a segmentation fault when the frame was processed. Besides adjusting the checks appropriately, we also tweaked the logic of the respective check. In the previous implementation, a frame was removed from the queue if its source or destination pointers pointed to a handler for a client registered with the ID of the disconnected client. Our understanding was that this could remove frames from a valid client handler registered with the ID in question, e.g. created by a reconnecting client. Thus, in our implementation the check will remove frames if and only if either the source or destination pointer points to the exact invalid handler, i.e. IDs will now be disregarded in this process.

Adding Packet Dumping, Uncovering Unscheduled Retransmits As a measure to help us analyse emulations and their results, we added a packet dumping mechanism to the MES. It allows files to be written in the popular PCAP dump file format, which can be viewed with programmes such as tcpdump [36] and Wireshark [64]. This feature revealed that sometimes frames were sent several times, even if the intended destination had already received the frame. As a response to this, we changed the internals of the queueing mechanisms. Instead of creating a new object for each received frame or retransmission, the Collision Extension will now handle a list of pointers to unused frame storages and hand them over, creating new ones whenever necessary, to the main programme which handles incoming frames. The main programme will populate the storage with the appropriate data and hand it over to the handler for the client that sent the respective frame, which will then add a unique ID and enqueue it in its client queue simply by appending the pointer. Moving the packet from the client queue to the main queue is again performed by moving the appropriate pointer from the client handler to the Collision Extension. For retransmissions, the Collision Extension will update a value indicating the transmission count and reorder the queue by the timestamp of the next transmission. Once a frame should no longer be retransmitted, the pointer to its storage will be moved to the list of unused frame storages from where it can be reintroduced to this cycle.

While we assume that the structure described in the previous paragraph should yield a significant performance benefit, our main goal was to ensure that every frame should only live exactly once in the MES and thus be uniquely identifiable. This allowed us to discover that the reasons for the unexpected retransmissions were in fact trivial: When processing the main queue, frames that had been marked as “sent” but remained in the queue to allow the detection of collisions would eventually be transmitted again and again until all copies had been purged from the queue because they could no longer interfere with any other transmission. While we did not formally evaluate the benefit of fixing this issue, in our tests the number of frames dumped during an emulation with a fixed version of the MES was about $\frac{2}{5}$ of the count obtained with an emulation of the same scenario but with the prior version, i.e. roughly $\frac{3}{5}$ of all transmissions appear to have been caused by the bug described in the previous paragraph.

Unpredictable Time-outs in ARP Cache A second issue we discovered using the packet dumping feature described above was that nodes would often fail to transmit frames because the transmitting node no longer had an ARP cache entry for the next hop on the route to the destination. This would result in the node querying for the next hop’s layer 2 address instead of trying to deliver the actual packet. In the observed instances, the query was quite often not successful, resulting in a failure to deliver packets on the given route, while in other instances the packet was delivered successfully through the given link. Since there appeared to be no predictable or reproducible pattern in this, we concluded that this affected the reproducibility of our emulations. Our first approach to solve the issue was enabling the OLSRd’s ARP caching plug-in, which creates ARP entries for nodes when receiving OLSR messages from them. This did not appear not to solve this issue for good so we wrote a component for the DBrES, our emulation system which will be discussed in the next section, which would add a fixed mapping for all nodes to each node’s ARP cache at the begin of each emulation. While the complexity and inflexibility of rolling out such a configuration in a real deployment renders this approach unsuitable in such an environment, nevertheless even there it would be a valid option and thus we consider the impact on our emulation’s realism negligible when compared against the improvement in reproducibility.

Socket Write Readiness Check Our experience with other programmes indicated that attempting to writing extensive amounts of data to a socket without a write readiness check may eventually fail, even if the socket is in blocking mode, which is supposed to prevent such issues. Thus, when we learned that the MES used blocking sockets but no write readiness checks on them, we introduced such a check. This turned out to solve the problem of clients frequently losing their connection to the MES, which had been our reason for switching to the extended version of the MES in the first place.

Since waiting indefinitely for all clients to become write ready would distort the model of the wireless channel, we decided to add a time-out to this mechanism, ignoring clients that were not ready to receive a frame within 100μs. We however included an exception for the destination of a unicast frame; there is no time limit for delivering a frame to a valid unicast destination. While a failed unicast transmission usually results in a retransmission, the Collision Extension is supposed to be the only point in the programme where these are modelled. Thus, discarding a frame at this point would result in the destination not receiving it without triggering a retransmission and eventually distort the model of the wireless channel.

Seeding the RiceanFading Oracle and Collision Extension While we already introduced the option of using a seed when configuring the RiceanFading oracle to allow setting it in a reproducible initial state, results obtained during the evaluations described in this thesis sometimes appeared to be too loosely correlated to assume reproducibility. We were unable to find any sources for that other than the ones already addressed. Thus we added a feature to define a seed for the Collision Extension, which would use a random number generator to simulate the back off determined by a wireless NIC and replaced the single random number generator used by each of these components by a batch of generators. While we generally assume that such a scenario should not occur in our emulations, this would ensure that the situation described below could not occur in them.

Assume that a random number generator R generates a sequence of random numbers r_i. When $(r_i, \textcircled{X})$ denotes "random number r_i has been drawn for node X" and given three nodes $\textcircled{A}$, $\textcircled{B}$ and $\textcircled{C}$, each of them requiring a random number to be drawn in a given interval, this sequence could be mapped to the given nodes as follows:

$\mathrm{Seq}_1 = (r_0, \textcircled{A}); (r_1, \textcircled{B}); (r_2, \textcircled{C}); (r_3, \textcircled{A}); (r_4, \textcircled{B}); (r_5, \textcircled{C}); ...$

Now assume that a non-controlled effect resulted in the first random number being drawn for node $\textcircled{A}$ only after the first random number for nodes $\textcircled{B}$ and $\textcircled{C}$ had already been drawn while the intervals at which numbers would be drawn would otherwise remain unchanged. This would result in:

$\mathrm{Seq}_2 = (r_0, \textcircled{B}); (r_1, \textcircled{C}); (r_2, \textcircled{A}); (r_3, \textcircled{B}); (r_4, \textcircled{C}); (r_5, \textcircled{A}); ...$

Obviously, our small change resulted in Seq_1 not only being different than Seq_2, but also not sharing any common subsequences. This will not occur if for each node numbers are drawn from their own generator, as intended by our change to the MES.

For simplicity, the seed configuration for each of these components consists of a comma-separated list of seeds and for each seed, a unique random number generator will be created and initialised with the given seed. The generator to use will be selected based on the ID of the node sending a frame or requiring a back off to be determined, but modulo the count of generators available. Therefore, as long as IDs are consecutive, when the number of seeds provided is as large or larger than the count of nodes, each node will have its own random number generator. More particularly, with this scheme the mapping between a node ID and a random number generator with a given seed is consistent among several executions of a given scenario without requiring any additional configuration.

5.1.3. The Diminutive Breadroll Emulation System

To counter a series of difficulties we experienced with emulations as well as to provide a means for consistent, reproducible emulations, we developed the Diminutive Breadroll Emulation System (DBrES).[2] It uses an object oriented approach and the popular Python scripting language

[2]The name is derived from the German proverb "Kleine Brötchen backen" or "to bake small breadrolls", suggesting that one should not try to master challenges above one's capabilities but rather stick to the challenges one can master. We chose the name in reference to an approach to building a highly sophisticated emulation system to replace a poor existing solution. The approach had to be abandoned because, due to the high aspirations, the complexity of the undertaking spiralled out of control. When developing DBrES, we learned from these lessons and provided a much less complex solution that was however capable of replacing

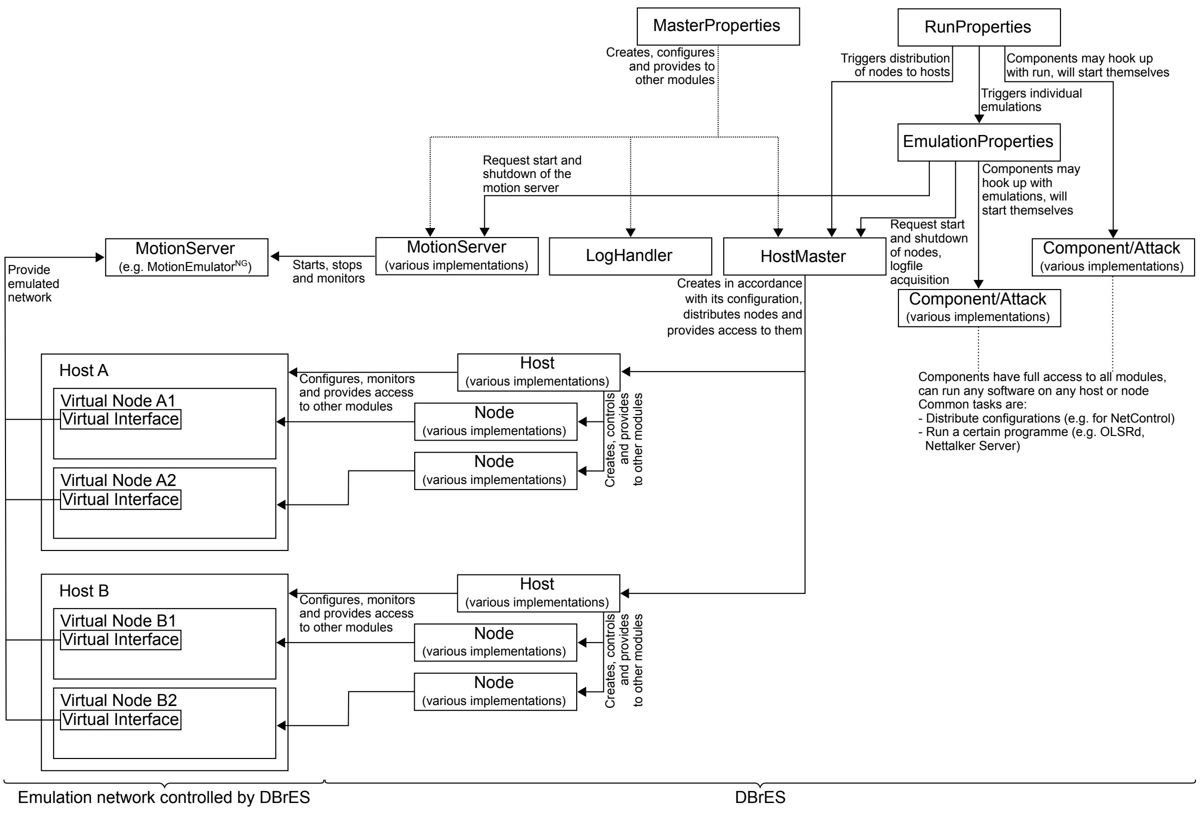

MasterProperties
Creates, configures and provides to other modules
RunProperties
Triggers distribution of nodes to hosts
Components may hook up with run, will start themselves
Triggers individual emulations
EmulationProperties
Components may hook up with emulations, will start themselves
Request start and shutdown of the motion server
Request start and shutdown of nodes, logfile acquisition
MotionServer (e.g. MotionEmulatorNG)
Starts, stops and monitors
MotionServer (various implementations)
LogHandler
HostMaster
Component/Attack (various implementations)
Provide emulated network
Creates in accordance with its configuration, distributes nodes and provides access to them
Components have full access to all modules, can run any software on any host or node
Common tasks are:
- Distribute configurations (e.g. for NetControl)
- Run a certain programme (e.g. OLSRd, Nettalker Server)
Host A
Virtual Node A1
Virtual Interface
Virtual Node A2
Configures, monitors and provides access to other modules
Host (various implementations)
Node (various implementations)
Creates, controls and provides to other modules
Host B
Virtual Node B1
Virtual Node B2
Emulation network controlled by DBrES
DBrES

Figure 5.2.: A simplified sketch of the DBrES structure. Each element of the emulation network (depicted on the left hand side) that should be controlled by DBrES has a representation therein. Modules marked with “various implementations” can have several implementations that provide a predefined interface to DBrES and its modules, including components. The latter provide most of its flexibility, allowing DBrES to start, monitor, stop any software, read or write files, e.g. to roll out a specific configuration, or do any other task the user is able to implement.

to be both easy to configure through a set of intuitive configuration files and allows adding functionality with little effort.

Figure 5.2 depicts the basic structure of DBrES. It uses a master configuration file to set up an instance of the MasterProperties class that provides some core functionality and settings that are usually consistent over a large set or all emulations. Among its options are general verbosity of logging and debug output, paths to directories containing files describing movement and event patterns or for storing results as well as the configuration for the MES or another kind of emulation server and the path to another configuration file that describes all hosts that may be used in emulations. The configuration of a host includes its type, its address, a description of the naming schemes, available VEs and capacity in terms of nodes that may be assigned to the host at a given time. We assume that both hosts and nodes use a POSIX-compliant operating system and that they are set up to allow password-free login using SSH and SCP from the host running DBrES, which can be achieved securely by using public key cryptography and SSH client certificates, but avoid further assumptions. In particular, we assume that Python is available on the system running DBrES but it is not required on any of the other systems

the previous system. Last but not least, the acronym “DBrES” can be read “debris” and is thus easy to remember.

involved.

Emulations of a similar kind, a “run” in our terms, are configured through a single configuration file read by an instance of the RunProperties class. Some of their properties, e.g. the total duration, verbosity of logging and number of retries until an emulation is considered to have failed irrecoverably, can only be configured through the “DEFAULT” section interpreted by the RunProperties instance. Individual emulations are set up through adding a configuration section that will be interpreted by an EmulationProperties instance. Theoretically, adding a section is everything necessary to ensure another emulation would be executed, however each section should provide the options that distinguish it from the other emulations in a given run. These will usually be the movement and event patterns to use but may include settings for other features, examples are seeds for the MES or specific configurations for extensions that we call components or attacks and explain in the next paragraph.

Most parts of DBrES can be set up to use different implementations, based on the user’s requirement or hardware set-up. Since Python does not provide abstract prototypes, we use a second-best approach of providing non-abstract prototypes that define all interfaces that would be used by DBrES. Thus, a programmer can provide a proper implementation by providing all the methods described in the respective prototype. Currently, the emulation server can be set up through such an implementation but also classes of hosts and nodes that should participate in emulations. We however also provide, in the same manner, the possibility of adding “components” or “attacks”. These are extensions that are supposed to be configurable through a run or emulation configuration. Depending on their set-up, they will be provided with a reference to a run’s or emulation’s configuration and informed about events such as an emulation being started, finished or aborted and may perform whichever actions are necessary for providing their own functionality. Components and attacks are equal in every aspect except that attacks will always be initialised after components, i.e. may check, refer to or modify a component’s configuration, if necessary. Components we currently provide allow rolling out configurations for RITA NetControl, starting a selected flavour of the OLSRd on all nodes, the Nettalker Server (NTS) on all server nodes or schedule events at improved precision. We also provide attack implementations to start and stop the sinkhole and wormhole implementations we created for this thesis.

A last to mention but nevertheless prominent feature of DBrES is that is it provides error checking at every level. During an emulation, each module is regularly polled to initiate a self-check, if an error is detected, the emulation will be aborted and started again unless the configured limit for retries has been hit. All of the components and attacks we implemented in the course of this thesis use this feature to provide early detection of faulty emulations.

Since it may not be possible or efficient to check for some errors on the fly, DBrES allows implementing “error checkers” that analyse the log files obtained from an emulation for known issues and will also initiate a retry, if an error has been found. Currently, we only provide an error checker for detecting the issue mentioned in Section 5.1.2 and described in further detail in [12].

5.2. Scenarios

5.2.1. Movement Patterns

Our approach evolves around organisations with security-related duties, e.g. rescue workers, policemen or infantry soldiers, operating in small but coordinated and cooperating groups of dismounted agents. We thus use movement patterns that were generated according to the Reference Point Group Model (RPGM) paradigm. RPGM refines the Random Waypoint Model (RWP) model which models the movement of independent nodes between randomly chosen waypoints at random speeds and possibly with random waiting periods before moving towards the next waypoint. In RPGM, nodes are organised in groups and each group has a virtual centre which moves in a RWP pattern. While the nodes still move independently from each other and using RWP patterns, they may only move within a certain radius from the aforementioned virtual centre. Whenever a node's distance would exceed the predefined radius, it will choose a new waypoint that will ensure it stays within the circle implicitly defined through the virtual centre and radius.

Since a German army infantry squad typically consists of ten soldiers, we chose RPGM groups of ten nodes each. We estimate that our emulation environment is currently capable of supporting 30 client nodes, so each scenario includes three groups of ten nodes each, moving on an area which is 1000m by 1000m square. Based on the assumption that nodes use commercial off-the-shelve (COTS) hardware with a maximum transmission range of 300m, we introduced similar conditions as in [10]:

1. The network may not partition, i.e. contain segments that are physically unable to communicate with each other given the assumed maximum transmission range, for more than 10% of a scenario's total duration of 1050 seconds.
2. Each node should reach at most $\frac{1}{2}$ of all nodes in the network at a time to ensure that multi hop routes will exist in the MANET.

Note that since we chose a group diameter of 300m, each node should be able to reach all other nodes in the same group with probability larger than 0; thus, the second condition basically implies that it would reach no more than five nodes from another or other groups. This would be expected for groups solving a common task and resembles the characteristics of deployed analogue radio equipment. The first condition may appear less obvious, if you consider however that nodes may be advised or ordered to adjust their behaviour in order to ensure that the connectivity of the network is ensured or restored, this becomes a valid assumption. Since concepts such as network centric operations imply that agents are able to exchange information rapidly (cf. e.g. [7]), rendering communication a central part of all operations, adapting the agents' behaviour in a way that ensures communication is possible most or all of the time even becomes an imperative.

Introducing the conditions described above implies that some scenarios that were chosen at random would not be considered since they would violate these conditions. Thus, conclusions based on emulations using these scenarios may fall victim to a loss of generality. [10] showed however that the ratio of randomly chosen scenarios meeting conditions similar to the above was actually quite high, requiring seven or less iterations for $\frac{3}{4}$ of 20 scenarios they generated. Table 5.1 provides an overview to the parameters used to generate the movement patterns described in this section.

Option	Setting
Area	1000m $\times$ 1000m
Duration	1050 seconds
Assumed maximum transmission range	300m
RPGM group count	3 groups
RPGM group size	10 nodes
RPGM group radius	150m
Minimum node speed	0.5 $\frac{m}{s}$
Maximum node speed	1.5 $\frac{m}{s}$
Maximum node pause duration	60 seconds
Additional Conditions	1. the network must be unpartitioned for at least 90% of the scenario's duration 2. each node is neighbour to at most $\frac{1}{2}$ of all nodes in the network

Table 5.1.: Settings used for generating the movement patterns for our evaluation.

5.2.2. Radio Propagation Model

Our choice of using the MES left us with choosing among the four oracles it provides to model the wireless channel. The "hub" oracle is not intended to provide a realistic model of a wireless channel and could thus be ruled out immediately. "TwoRayGround" implements a simple distance based on/off-state wireless channel which is neither considered realistic nor does it provide a base for executing a forged link quality attack since link qualities will generally be either 0 or 1. The "Fading" oracle provides a fixed probability curve, determining the probability of a frame being received for a given distance between sender and receiver, degrading the quality of the link over distance. While this provides a more realistic model of the channel, it does not model any small scale effects. This is provided by the "RiceanFading" oracle, which uses a probabilistic model of the wireless channel and is thus both the most realistic model provided by the MES and the obvious choice for our evaluations.

Since we assume the use of COTS hardware, we provided a configuration for the RiceanFading oracle which resembles a commercial wireless NIC. We chose the same device, the "Cisco Aironet 802.11a/b/g Wireless CardBus Adapter", as in [10] but assume that it was set up to use a different mode. In [10], the IEEE 802.11a mode was used which is however seldomly used in practical deployments and has been superseded by the IEEE 802.11 b, g and n standards. We thus use the characteristics provided by the manufacturer when the device is set to the IEEE 802.11g mode at a rate of 11MBit/s.

The NIC may however also be operated at several power levels to adjust it to specific environment conditions, restraints in energy consumption or regulatory requirements. While the latter may not apply to military contexts, both energy consumption and environment conditions may play a role there. More particularly, nodes may prefer to reduce the transmission power not only to reduce energy consumption but also to reduce the radio footprint of deployed troops. Based on the scenarios described in the previous section, we decided that the sweet spot for trading off between energy consumption, radio footprint and achievable transmission range should be where transmissions over a range of 300m still have a good chance of reaching their destination.

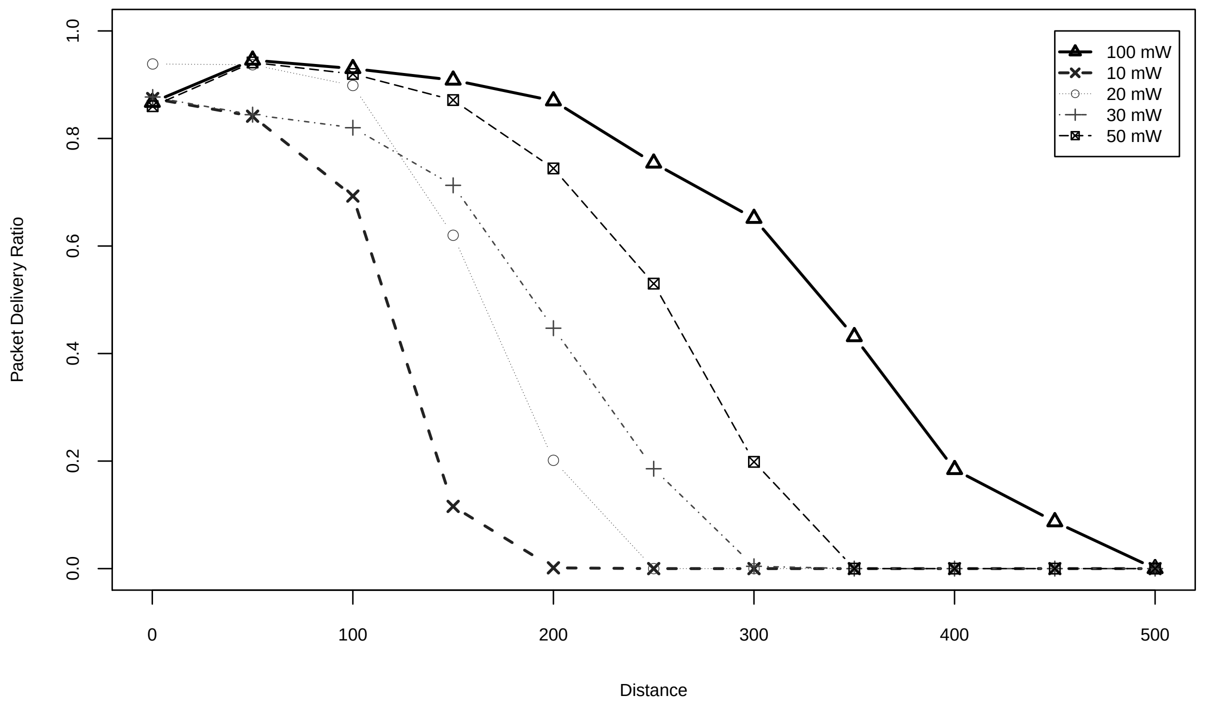


Figure 5.3.: The packet delivery ratios achieved with different configurations of the MES at a distance increased in steps of 50m. Each curve reflects the packet delivery ratio achieved for a given configuration and at a distance increasing in steps of 50m. While each configuration models the Cisco Aironet 802.11a/b/g Wireless CardBus Adapter wireless NIC set to a rate of 11MBit/s in IEEE 802.11g mode and with retransmissions disabled, different transmission power levels were used in the individual configurations. Thus, each curve reflects the PDR achieved with the respective power levels given on the top right hand side of the graph.

To evaluate the chance of delivering a frame at a given distance and power level, we ran simple emulations with the respective configuration but turned off retransmissions by the MAC layer modelled by the MES. In the emulations, one node would stay put for the whole duration while another one would start right next to it but move away from the other node in steps of 50m. While stationary, the node would send 700 packets to the completely immobile other node, since there would not be any retransmissions, the packet delivery ratio for these bursts would be equal to the rate of packets delivered through the modelled wireless channel at the given distance and thus provide a basis for estimating the packet delivery ratio to be expected at a given distance.

Figure 5.3 shows the PDR obtained at several power levels but otherwise unchanged settings. It indicates that at a distance of 200m, there will be almost no successfully delivered frames when the transmission power is set to 10mW while with 20mW the PDR is still just above 20%. For both settings however, the PDR drops to zero at 250m. With a transmission power of 30mW, three frames would still be received at a distance of 300m, i.e. the respective PDR would be just above 0.4%, too low to assume that transmissions at that distance have a fair chance of arriving at their destination. At the next higher setting, 50mW, just below 20% of all transmitted frames were successfully received, while at 350m no single frame arrived. Since with the remaining setting of 100mW transmissions still had a fair chance of success even at 400m and 450m, we decided that setting the transmission power to 50mW yielded

General MotionEmulator settings	
Oracle	RiceanFading

RiceanFading oracle settings	
Transmit power	50mW
Receive threshold	-90dBm
Carrier sense threshold	-95dBm
Coherence	disabled
Reference distance	300m
Reference path-loss	108
Path-loss exponent	2.3
Ricean K	4

Collision Extension settings	
Data rate	11 MBit/s
Signal to noise ratio threshold	10
Maximum retransmissions	4

Table 5.2.: Settings for the wireless channel model of the MotionEmulator^{NG}.

a good approximation of our desired maximum transmission range of 300m. The resulting configuration for the RiceanFading oracle and the MES' Collision Extension is provided in table 5.2. Note that, other than for the evaluations concerning the maximum transmission range described in this paragraph, retransmissions are enabled in these settings.

5.2.3. Network Traffic

Traffic Model

The network traffic in our emulations is generated by several components. All nodes run the RITA client software, a single node, designated as a fully equipped node in RITA terms, is supposed to be equipped with a larger battery and more processing power and thus runs the respective additional programmes. The RITA client programmes generate traffic by sending reports to the fully equipped node in regular intervals.

Additional non-user traffic is generated by the OLSRd providing the routing in the emulated MANET, including reports generated by the implementation of the client components of both the ToGBAD and ToGBAD-LQ concepts. While we expect the non-user traffic to be roughly consistent among several replications of a given scenario, manipulating these components to produce more predictable traffic would not only require a significant amount of effort but would also mean effectively modifying the protocols they implement. Thus, results obtained with such modifications would not necessarily coincide with results obtained with the original implementations and this option was ruled out for this thesis.

Generally, personal digital devices would replace analogue personal radio equipment which is currently deployed with the services we consider in this thesis. Therefore, they would not be perceived as an improvement, if they did not provide the service, i.e. voice communication, that the existing equipment provides. Our emulations should thus include a model for traffic

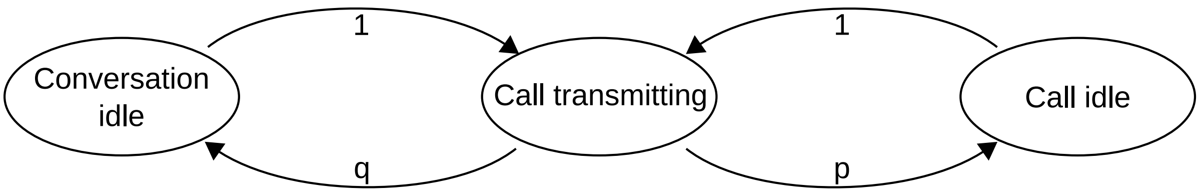


Figure 5.4.: The three state semi-Markov model for radio conversations proposed in [8].

Settings for modelling radio conversations	
p	0.7
q	0.3
Conversation idle duration (log normal distribution)	$\mu = 1.390844, \sigma = 1.628825$
Call transmitting duration (log normal distribution)	$\mu = 0.4438584, \sigma = 0.7766512$
Call idle duration (gamma distribution)	$\alpha = 1.530391, \beta = 1.60653$

Table 5.3.: Configuration used when generating radio conversation patterns for this thesis. The given numbers were suggested in [8].

generated by voice communication. [8] introduced a model for radio conversations based on a three state semi-Markov model. While the model was derived form the observation of exercising rescue services, we assume that since the radio communication of these services follow similar rules and regulations as in other services, e.g. the armed forces, the model remains valid in these settings as well.

Figure 5.4 shows the basic model, a given conversation starts in the idle state, but must change to "transmitting" state. When doing so, our implementation of the model chooses two nodes uniformly at random that engage in a conversation. From the transmission state, a transition to the "call idle" state will take place with probability p while with probability $q = 1 - p$ the state will change back to "conversation idle". When returning from the former to the transmission state, the node which did not take turns the last time the model was in the transmitting state will take turns in transmitting. In the latter transition, our implementation would forget the nodes participating in a conversation and choose new nodes with the next transition to the transmitting state. Note that the model will remain in each state for a random amount of time, depending on the model's parametrisation, reflecting a time frame during which no conversation takes place, one peer is talking while the other is listening or none of the two is talking. Table 5.3 shows the settings we used when generating our patterns.

Given the setting defined in the two previous sections, i.e. that we have three groups of ten nodes each, we planned to model both conversations within each particular group as well as conversations between a command and control group consisting of one designated node for each of the groups. However, since the maximum distance between two nodes in a given group is equal to the maximum transmission range implied by the radio propagation model, there would be little, if any, packets travelling more than a single hop until they would reach their destination. Hence, there should be little or no impact of a routing attack on this communication channel, indicating that analysing such an attack's impact on the channel would not provide any predicate on the effectiveness of the attack. Since modelling this channel would still create extra load on our emulation environment, we decided to avoid possible side effects created by the additional resource consumption by not modelling this channel but restricted ourselves to modelling the command and control channel.

In addition to that we want to point out that the OLSRd that we use in our emulation environment does not provide any multicast routing. Thus, we had to model each multicast transmission by a batch of unicast transmissions, one for each recipient. This however drastically increases the count of frames sent for each transmission, particularly for transmissions within a group where we have a large number of recipients which would typically all be in the sender's transmission range, significantly aggravating the impact of modelling communications within groups and thus supporting the decision laid out in the previous paragraph not to model this part of the expected network traffic.

Implementation

The model described, and thus our implementation thereof, in the previous section defines radio conversations as sequences of nodes transmitting or not transmitting. For producing these sequences in our emulations, we opted for using the "melp-generator" we had written for similar tasks before. It consists of two small programmes, the "melp-sender" sends UDP packets of a given length and at a predefined interval and duration while the "melp-receiver" listens on one or several ports for incoming packets and writes a log file including all successfully received packets. To start the respective transmissions, we wrote a component for DBrES which starts the melp-receiver programme at executes the "melp-sender" programme. The programme will send packets at an interval of 67.5ms with 21 bytes of payload each.

The interval and payload were chosen to coincide with three frames of 54 bits (6.75 bytes) each, i.e. a total of $\lceil 3 \cdot 6.75 \rceil = \lceil 20.25 \rceil = 21$ bytes of payload, that would be created by NATO's STANAG 4591 MELPe codec when using it at a bit rate of 2400 bit/s [2]. Since each MELPe frame would encode 22.5ms of voice, packets containing three of them should be generated every 67.5ms.

As there is no actual voice data to transmit, the payload of each packet consists of two unsigned 32 bit integers reflecting the microsecond precision timestamp of the packet's creation followed by another 32 bit unsigned integer which is simply increased for each packet in a burst, i.e. acts as a sequence number. The remaining 13 bytes are set to 0.

5.3. Obtaining Consistent Results

Despite the extensive effort, as described in Section 5.1.2, that we invested in providing a reliable emulation environment, results that were obtained from several executions of a given scenario often differed by a significant degree. While we assume that the individual emulations were consistent in that the environment provided a fair model of a live environment, for assessing the impact of the sinkhole attack as described in Section 6.1, we had to draw conclusions correlating executions of a scenario without and with an attack. If the effective model would differ significantly between two such executions, false conclusions might have been drawn from comparing them.

Since choosing, setting up and testing an alternative environment was not possible within the intended time frame, we had to define reasonable constraints for assuming that an emulation could be considered providing reproducible results. Anticipating the metrics used to measure the impact of the attacks, described in more detail in Sections 6.1.1 and 6.2.1, we were able to

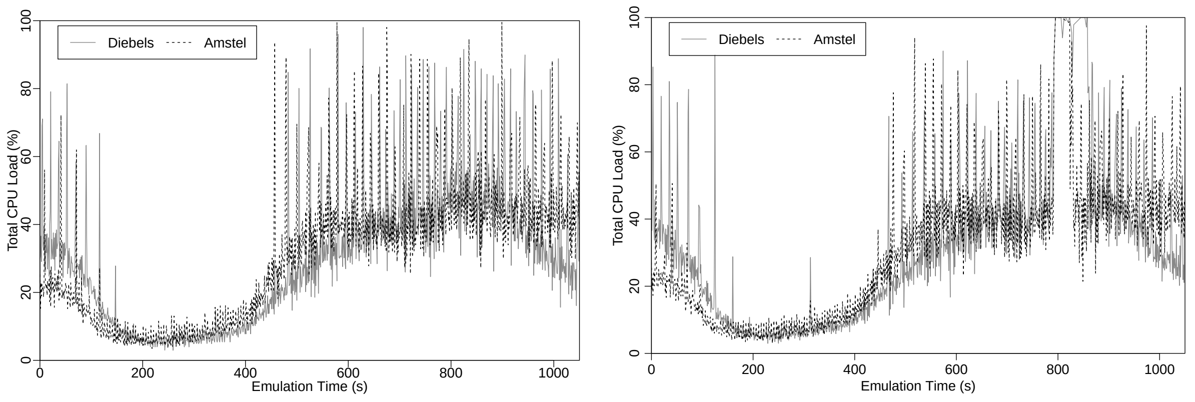


Figure 5.5.: Plots of the CPU utilisation of both hosts used in two emulations using the exact same parameters. While the emulation on the left hand side did not show any anomalies, a peak in CPU utilisation at about second 800 on the right hand plot resulted in a degraded PDR for transmissions started in the respective time frame. Note that while the graph may appear conspicuous between seconds 200 and 400, we were able to verify that CPU utilisation just reflects the low neighbourship grades for the nodes in the network during the respective time frame.

determine the impact of a wormhole attack without relying on results from an execution of a scenario without an attack but our metric for the sinkhole attack relied on comparing the PDR obtained in a scenario without and with an attack. Therefore, irregularities concerning the PDR in such emulations would have had an impact on our conclusions.

To provide more reliable results for our evaluations, we imposed two restrictions on the potential results. After discovering that the CPU utilisation in our emulation environment would eventually be exhausted for suspended periods, we compared the PDR of the bursts that occurred in such a period obtained in an emulation where resources had been exhausted against the PDR obtained in emulations of the same scenario where no peak occurred in the time frame in question. It appeared that while not for all bursts in question at least some would exhibit a significantly lower PDR in the emulations where CPU resources had been exhausted. Thus, we decided that an emulation's results should be discarded when at least one computer's CPU utilisation had been at 100% for two or more seconds.

Figure 5.5 shows a graph of the CPU utilisation for both computers participating in two emulations of the same scenario. While they appear to closely resemble each other at first sight, the graph on the right hand side shows a period of about 70 seconds starting at about second 800 during which both machines exhausted their processing power for a sustained period. When analysing the PDR obtained in these emulations, we discovered that for a burst which would start shortly after second 800, the burst's PDR in the emulation that the left graph refers to was close to 100% as for most of the other iterations. For the other emulation, the PDR was at about 50%, indicating that exhaustion of processing power could affect the PDR obtained in the affected emulation.

Note that Figure 5.5 also reflects a phase of very low cpu utilisation at about seconds 200 to 400. While this may appear conspicuous, we were able to verify that in the given scenario the RPGM groups were particularly far away from each other, leading to a low neighbourship degree for most nodes (Appendix A.1 provides an overview to the movement pattern for the respective scenario 11). Thus, most of all routing messages would be received and processed by

Scenario	PDR burst deviation ratio	
	without attack	with attack
0	11.57%	12.40%
1	12.20%	19.51%
2	6.84%	18.80%
3	7.19%	25.90%
4	24.19%	25.00%
5	8.03%	21.90%
7	6.15%	13.85%
8	5.56%	26.67%
9	10.56%	29.58%
10	7.64%	34.03%
11	9.65%	21.05%
14	11.71%	18.92%
15	10.00%	22.50%
18	18.00%	24.00%
19	12.93%	28.45%

Table 5.4.: This table shows the ratio of per burst PDRs deviating by more than 0.1 from the respective median determined over 15 samples per scenario. Note that some scenarios were not included since we were not able to produce the desired number of samples for them.

fewer nodes and contain fewer entries, resulting in the significantly reduced need for processing power reflected by the graphs.

Analysing existing results revealed that while roughly $\frac{2}{3}$ of our emulations without attacks would pass this requirement, less than 5% of the existing results of emulations with attacks would. Given that, we wrote an extension to DBrES that executes this check on line, i.e. aborts and restarts emulations that would not pass the criterion, but still our emulation environment would only produce about one valid emulation per full day of running emulations.

To counter that we took two additional measures. First, we added a third computer to our emulation environment, distributing the emulation's load on more computers reduced the total on each individual machine. However, the emulations with attack for five particular scenarios still exhibited an extraordinary rate of violating the CPU load criterion, i.e. resulted in CPU load peaking for two or more seconds, while this was rare for any of the other scenarios. Thus, we removed the former scenarios from our input set, boosting our success rate.

Having a high success rate was important for implementing our second restriction. Given that the per burst PDR would remain volatile despite the efforts described above, we had to define a reference for what should be considered the most reproducible PDR for each burst. Since the median is considered to be particularly robust against outliers, we decided that the median of each burst's PDR, obtained from 15 emulations for each scenario should provide a meaningful point of reference. However, we would have to pick particular emulation results for each scenario for our analysis, so we decided to classify deviations from said median of up to 0.1 in absolute terms as insignificant and deviations of more than 0.1 as significant. Ideally, a selected emulation result would not display any significant deviations, since this would however never be the case for any of the emulations incorporated in the median, we chose the result which exhibited the least significant deviations.

Table 5.4 provides an overview to the ratios of significant deviations achieved for the selected results. It reflects our earlier observation that emulations without attacks appear to be more reproducible while still there would not be any without burst PDRs deviating significantly from the median. With attacks, no single result exhibits a single digit ratio of significant deviations, for scenario 10 even the best performing emulation results deviate from the median's PDR by more than 34%. We included detailed graphs of the per burst PDRs for all scenarios in Appendix B.

5.4. Summary

This chapter started with an introduction of our emulation environment in Section 5.1, having a particular focus on the two main software components, ME in Section 5.1.2 and DBrES in Section 5.1.3. An extensive amount of time and effort was consumed by our attempt to provide a stable environment for producing reproducible results based on the MES. While we were able to fix a significant amount of bugs and to generally improve reproducibility, the environment at our hands was still far from satisfying.

Section 5.2 described the scenarios we created for the evaluations described in chapters 6 and 7. We will use RPGM movement patterns with three groups of ten nodes each, resembling a German army infantry squad. One node from each squad will be designated as a squad leader, communicating with the other squad leaders using a simulation of MELPe based software defined radio. The radio propagation model resembles a COTS IEEE 802.11 wireless device providing a desired maximum transmission range of 300m.

Despite the aforementioned efforts to improve our emulation environment, it did not meet our standards concerning the reproducibility and thus reliability of the results obtained with it. To be able to produce results with an at least acceptable level of reproducibility, we defined the criteria described in Section 5.3. Since we were unable to meet these criteria for five scenarios, we removed them from the population for the evaluations described in Chapters 6 and 7, leaving us with 15 scenarios.

Chapter 6

Evaluation of the Impact of MANET Routing Attacks

The MANET research community has directed a significant amount of effort at developing approaches that counter or detect the sinkhole and wormhole attacks. However, most research is based on simulations with few, if any, results obtained in emulations or field tests. While the primary focus of this thesis is the detection of these attacks, we also want to be able to assess their impact. Thus, in this chapter we will describe metrics that reflect the influence of the respective attack, our implementation and finally what impact we were able to achieve in our emulations.

The chapter is divided in two main sections. The first section covers the sinkhole attack while the second section is devoted to our implementation and evaluation of the wormhole attack. Following the two main sections, we close this chapter with a brief summary in Section 6.3.

6.1. The Sinkhole Attack

We begin with defining the metrics we want to apply for the sinkhole attack in Section 6.1.1, followed by a description of our implementation of this attack. Section 6.1.3 reflects the configuration we used for executing the attack in our emulations and finally Section 6.1.4 gives an overview to the impact achieved.

6.1.1. Metrics

An attacker perpetrating a sinkhole attack generally has the goal to attract traffic, i.e. the more packets are routed through it, the more successful its attack proved to be. In a MANET using the OLSR routing protocol, an attacker may fall victim to its own success as indicated in Section 2.3.1. When the attacker tries to deliver a packet destined to a node that the attacker manipulated the routing for, its neighbours may have been convinced that the best route to that destination is through the attacker and would thus simply forward the packet back to the attacker. In fact, that the attacker has been chosen to forward the given packet shows that at least one neighbour already determined the attacker to be the next hop on the best route to the destination. While this could be avoided under certain circumstances, such as the availability of collaborating nodes or the destination being in the attacker's transmission range, the attacker will generally not be able to deliver packets to arbitrary destinations when using the OLSR protocol. It should thus drop packets that were attracted by its manipulation of the routing.

Thus, as indicated in the previous paragraph, we should expect a sinkhole attacker to drop packets, i.e. execute what we termed a blackhole attack in Section 2.3.1, in the kind of networks considered for our evaluation. The more successful it was in attracting routes, the larger the share of packets will be which will be routed through the attacker and eventually be dropped by it. To estimate the impact of an attacker, we may thus compare the PDR achieved in an emulation without any attacks against another emulation of the same scenario but with a sinkhole attack. For this purpose, we only consider the payload traffic generated as described in Section 5.2.3.

6.1.2. Implementation

As a part of the project described in [12], we created an implementation of the sinkhole attack as a part of a plug-in for the OLSRd. The plug-in implements the local part of the ToGBAD-LQ mechanism described in Section 4.2; we provide further details on that part of the implementation in Section 7.1.1. Since the plug-in implements a customised OLSR protocol, introducing link qualities using the ETX metric and the challenge-response mechanism required to implement ToGBAD-LQ, existing implementations of the sinkhole attack could not be used, since their messages would simply be ignored by an OLSRd running our plug-in.

Therefore, we had decided to add the capability to execute a sinkhole attack to our own plug-in. To avoid accidental use, the plug-in has to be compiled in a special "attacker" mode or otherwise it will not contain any of the malicious code. The plug-in will however not execute an attack immediately, but will behave like a normal node until receiving an "attacker control message" that sets it up to perpetrate an attack, allowing to configure the attack dynamically without losing the internal state obtained during the non-attack phase. Also, the attacker will always execute the regular neighbourhood and topology discovery mechanisms and forging messages is executed by injecting the malicious data when generating them. Thus, the local database of an attacker should generally not be affected by its own attack.

Our implementation allows choosing among several modes of operation for the sinkhole attack. Since the basic plug-in provides support for link qualities to the OLSRd, the sinkhole attack inevitably implies forging link qualities. Assume an attacker would forge a link but not its link qualities. Since the link did not exist, each of the unidirectional link qualities would be 0 and thus the respective ETX value would be infinity and the link would never be used. Therefore, forging a link without any link qualities would constitute a minor waste of bandwidth and processing time but not an attack. Each mode of attack thus implies setting link qualities to a value determined by the attacker.

Modes of operation are generally:

- Forge neighbourships to all addresses in a given range
- Forge neighbourships to a given count of addresses in a given range, excluding existing neighbourships
- Forge link qualities for existing neighbourships

As pointed out above, the first two modes imply forging the link qualities to the forged neighbours. Also, these two modes are mutually exclusive while the latter would not qualify as a sinkhole attack in strict terms but could be combined with any of the two. Note that in each mode, link qualities to existing neighbours will not be forged unless the latter mode is active as well, even when addresses in the configured range overlap with those of the attacker's neighbours.

To start the attack in our emulation environment, we use three components building up on each other. The first part is an attack component for DBrES which allows configuring the attack through a DBrES configuration file. It will take care of starting the OLSRd with the attacker version rather than the regular version of the plug-in and add an event to DBrES's event system that calls the second component, the "start-challenge-attacker" shell script located in each node. The script receives the parameters for the attack and calls the third and last component, the "olsrd_lq_challenge_attacker_controller" programme. While the latter simply sends an attacker control message required to set the plug-in up to execute the given attack and then exists (cf. Appendix D for details on the control message format), the former, after calling the latter, simply waits for the begin of the attack time frame. Once it is reached, it will add a rule to the node's iptables firewall to drop any packets the node is supposed to forward and again wait until the end of the attack time frame is reached before deleting the rule again.

Note that the attacker plug-in will only execute a pure sinkhole attack, i.e. it will create forged neighbourships and link qualities but does by itself not ensure that packets would be dropped by the attacking node. As described in Section 6.1.1, this is however the expected behaviour of a sinkhole attacker in a network using the OLSR routing protocol. Hence, we require the aforementioned shell script to create the respective firewall rules to execute the attack in the intended fashion. Finally, we use an attack component to interface the attack with DBrES, allowing it to be started and stopped automatically.

6.1.3. Configuration in Our Evaluation

[23] presented an analysis of the impact of a blackhole in simulated MANETs, forging neighbourships to an additional $\frac{1}{3}$, $\frac{2}{3}$ or all nodes in the network. Their results indicated that forging neighbourships to more than $\frac{2}{3}$ of the nodes in the network yielded no further benefit in degrading the PDR in their scenarios while potentially facilitating the detection of the attack. Based on these findings, we decided that the attacker should forge neighbourships to $\frac{2}{3}$ of the nodes in the network, we however had to cope with that the attacker might be a neighbour to more than $\frac{1}{3}$ of the nodes in the network and would thus not be able to forge neighbourships to the implied count of 20 nodes. To resolve this, we decided that the attacker should forge 20 neighbourships, where the addresses of the forged neighbours will be taken from the address range of the nodes present in the network. If the respective address space should be exhausted, the attacker will use addresses of non-existing nodes to reach the desired count of 20 forged neighbourships.

We want to point out that while an intrusion detection system could exploit the fact that the attacker will claim neighbourship to a non-existing neighbour, ToGBAD does currently not support any such a check, so this will not provide a direct advantage to ToGBAD. Also, an attacker will usually not be able to determine all valid addresses in a network unless it was able to infiltrate a significant portion of its nodes. At that point however, it may even no longer be in the interest of the attacker to perpetrate a routing attack since it might conflict with more sophisticated attacks that would be enabled by such a massive intrusion. If on the other hand, the attacker's access is limited to a few or even a single node, it will have to make a guess for

DBrES CRETX attack configuration	
Attack time frame	Between seconds 100 and 500 of the given emulation
LQ values for forged neighbours	1 (maximum) for both unidirectional links
LQ values for real neighbours	1 (maximum) for both unidirectional links
Draw forged neighbours from	Addresses 10.0.0.1 to 10.0.0.50
Count of forged neighbours	20
Chosen attacker	

Scenario	0	1	2	3	4	5	6	7	8	9
Node (ID)	3	19	23	2	25	19	23	30	9	13
Scenario	10	11	12	13	14	15	16	17	18	19
Node (ID)	7	29	10	15	22	3	25	15	22	3

Table 6.1.: The configuration of the DBrES attack component for the challenge response ETX (CRETX) plug-in.

the address range which addresses for forged neighbourships should be drawn from. Such a guess might look exactly like the one implied by our configuration.

The next choice an attacker would have to make is for the link qualities to claim for forged neighbourships. While using a value below the maximum could render the detection of the attack more difficult, it would also reduce its impact. Since a successful attack might help rendering its detection more difficult, trading a possibly lower risk of detection against a certain loss of impact will generally not be in the interest of an attacker, unless it has detailed information on the IDSs it is up against and their configuration. Our attacker will thus claim the maximum link quality value of 1 for both unidirectional links between itself and any of the forged neighbours.

Finally, we have to consider whether link qualities between the attacker and its real neighbours should be forged. Given the argument above, i.e. that the attacker should seek to maximise its impact, it appears wise to forge these link qualities as well and claim the maximum value of 1. If an attacker was aware of the presence of an implementation of ToGBAD-LQ, it might adjust its behaviour, i.e. not forge the link qualities to its neighbours or forge them by a degree which appears unlikely to trigger an alarm. Given that the attacker would not be aware of such a mechanism, it would be unlikely to make that trade-off and just claim perfect links to the existing neighbours as well.

For each emulation with an attack, we chose a single attacking node uniformly at random from all nodes except for the nodes participating in the command and control channel described in Section 5.2.3. Choosing one of those nodes would have implied that the attacker would either suppress the transmissions generated by the node or leave them unaffected where the first would improve the success ratio of the attack without requiring the attacker to actually affect any node's routing table while the latter could serve to reduce it, i.e. discriminate against the attacker. A third option, ignoring the burst generated by the attacker node, would reduce the population when determining the PDR, which would no longer reflect the state of the network in the respective time frames as it would do in the PDR that the value should be compared against. Since none of these options appeared favourable, we decided that the best solution was to exclude the respective nodes from the population we would choose the attacker from.

The attacker chosen in the fashion described in the previous paragraph would then perpetrate an attack starting at second 100 of the given emulation and lasting for 400 seconds. Table 6.1 provides an overview to the configuration presented in this section and the nodes chosen for

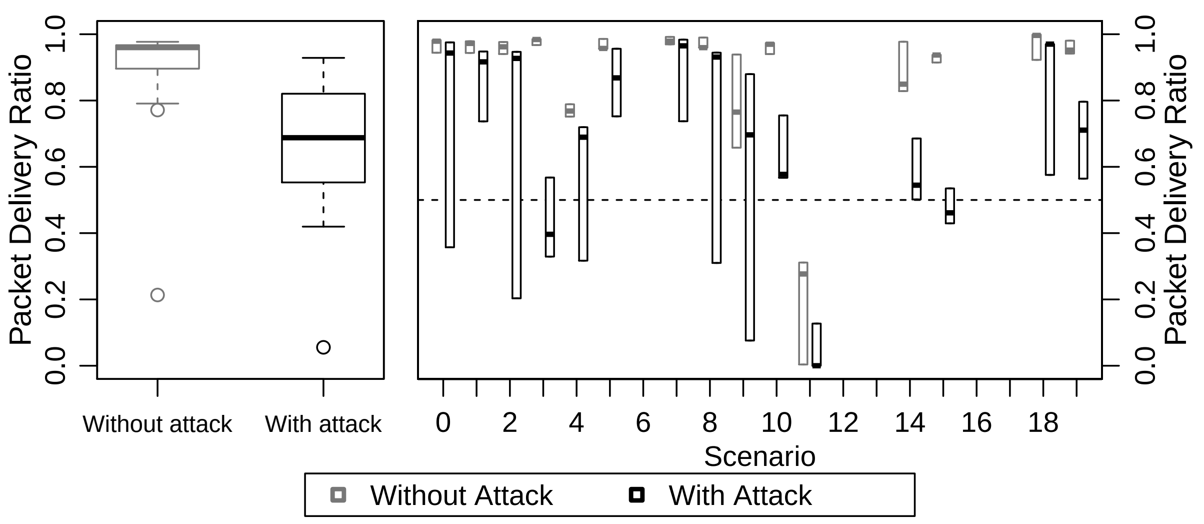


Figure 6.1.: This figure displays the PDR in the attack time frame both for emulations without and with a sinkhole attack being perpetrated. On the left hand side, it displays a summary of the total PDRs achieved during the attack time frame, while the panel to the right shows the PDRs obtained for each node and in each scenario. Here, the top and bottom of a box represents the highest or lowest PDR obtained for a node while the remaining node's PDR is indicated by a thick line with in the box. A dashed line indicates at which PDR recipients will no longer be able to understand the content of a transmission. In both graphs, grey boxes refer to results obtained without an attack while black boxes reflect the results in an emulation where an attack took place.

each scenario.

6.1.4. Results

The results obtained through our evaluation are pretty clear. The sinkhole attack reduced the total PDR in all scenarios, by about 30% for most but not all of them, as indicated by the box plots on the left hand side of Figure 6.1. For further discussion, we will however refer to the detailed graph on the right hand side of Figure 6.1. Since we are using an unusual representation, we will start out by describing how to read that graph. For each scenario, as indicated by the position on the x-axis, the graph contains a representation of the PDRs obtained both in an emulation without and with an attack. The former is represented by a box with grey outlines while the latter is printed in black. Since there are three nodes communicating, each box reflects the value for three nodes. With this information, reading the boxes becomes simple, the upper end indicates the largest PDR obtained for a node while the lower end refers to the lowest PDR, leaving us with the third node's PDR, which is indicated by a thick line inside the box. [19, 30] indicate that transmissions using the MELPe codec become inaudible when about 50% or more of their frames are lost. Thus a dashed line reflects that threshold or that we expect recipients to be no longer able to understand the peer's message, i.e. communication would become impossible, if the PDR drops below it.

Figure 6.1 indicates that in eight scenarios one node was affected significantly more than the others. While the quality of their transmissions would still have decreased, the third node would no longer reach the 50% quorum in five of these cases (scenarios 0, 2, 4, 8 and 9). Correlating these results with the movement patterns as depicted in Appendix A.1 indicates that this is caused by the group in which the attacker was located being particularly far away from the other groups. In this situation, legitimate routes will traverse more links which may exhibit lower link qualities, making the routes offered by the attacker more attractive in comparison.

In the remaining seven scenarios, the node's PDRs were affected more evenly. While in Scenarios 3 and 15 this enabled the attacker to reduce two node's PDRs below the 50% threshold, this generally does not work in favour of the attacker. In Scenario 11, it reduces two node's PDRs to 0 but since all node's PDRs were already well below 50% even without an attack, this does not provide a benefit to the attacker. When including this scenario, the attacker does not reach its goal in five out of seven attempts.

Analysing the respective movement patterns indicates that if two groups, including the attacker's, are located close to each other, the attacker has a good chance of influencing the routing to the third group, but is unlikely to attract significant amounts of traffic between these groups. Scenario 3 deserves some extra attention. Here, the groups are clearly separated during the attack time frame and the attacker's group serves as a bridge between the two other groups. Since the attacker claims to have the best links to the node's in the other groups, it has a good chance of being chosen as the next hop not only for the packets sent to those nodes by members of its own group, but also for packets that should be relayed through it. This is reflected by the significantly decreased PDR portrayed for Scenario 3 in Figure 6.1.

Summarising, the sinkhole attacker generally benefits if nodes are scattered over a larger terrain. This comes at no surprise since under these circumstances accumulated cost for routes will be larger. On the other hand, only in seven out of 15 scenarios it reached its goal of suppressing at least one node's ability to send audible transmissions to the other nodes. Also, its options were limited to dropping the attracted traffic, i.e. it would not be able to build up momentum or wait for a particular moment until affecting the quality of service in the network. Therefore, both success probability and capabilities gained through the sinkhole attack are limited. We would like to point out however, that a sophisticated attacker may use the topology information provided through the OLSR routing protocol to estimate the potential impact of its attack and thus launch it only when the topology promises most impact.

6.2. The Wormhole Attack

In this section, we will focus on the wormhole attack. Following the structure of Section 6.1, we will start out with introducing the metric we will use to measure the impact of the attack in Section 6.2.1, followed by a detailed description of the wormhole implementation we wrote for this thesis in Section 6.2.2. Thereafter, Section 6.2.3 provides the details of the set-up in our emulations. Finally, we describe the results concerning the attack's impact in Section 6.2.4.

6.2.1. Metrics

As described in Section 2.3.3, the wormhole attack is aimed at attracting traffic, allowing the attacker to analyse it or discard packets at will. Thus, the larger the portion of the traffic tunnelled through the wormhole, the more successful the attack turned out to be. Since the attack has no other effect that could be measured directly, our metric should be the ratio of the traffic tunnelled by the wormhole.

To be able to determine the aforementioned ratio, we would have to add a method for tracking tunnelled packets. While we implemented a packet dumping feature for the MES, as described in Section 5.1.2, our wormhole would remain transparent to the MES and thus this would not help in determining whether a dumped transmission was in fact tunnelled or not. Since, other than in simulations, it is not possible to add information to a packet in emulations without actually modifying it, we would have to do just that.

However, to detect a modification, we need to be able to determine whether or not a packet has been modified in transit. For arbitrary protocols, this is impossible to achieve without storing the respective packet and having the means to determine whether it arrived at its destination. While the latter would be possible through our MES packet dump file, the former would have posed a significant challenge. Even more, modifying packets might invalidate their payload, since that was not the assumed goal of our attacker, we wanted to avoid the implied protocol breakage.

On the other hand, our model of the payload traffic, as described in Section 5.2.3, implies the transmission of packets with a predictable payload. The respective receiver programme not only logs the payload of incoming packets, but will also not be affected by any modifications to it. Since we assume that the respective traffic is representative for the whole traffic in the network, modifying these packets and considering their ratio of tunnelled packets solves the challenges laid out above.

6.2.2. Implementation

Since there was no openly available implementation of the wormhole attack, we had to create or own to be able to determine the impact of this type attack. In this section we provide a brief overview on the Bonner Hole (BoHo),[1] our implementation of the wormhole attack.

The BoHo captures frames from a NIC attached to an attacked network, using the libPCAP packet capturing library [36], and forwards them to a user defined list of endpoints. Upon receiving a forwarded frame, an endpoint will try to retransmit the frame on the NIC it would also capture frames from. There is no error correction, i.e. the wormhole implementation operates in a “best effort” mode which tries to ensure fast retransmission, but recovers from errors simply by tearing down and re-establishing its connection to the failed endpoint.

Also, our implementation assumes that an out-of-band channel is available for connections to other wormhole endpoints. As summarised in Section 2.3.3, we see no application for in-band wormholes. We want to point out however that it is possible to use libPCAP filter statements with the BoHo which could be used to ensure frames containing forwarded packets would not be retransmitted. However, as described in [44], in-band wormholes require a relay which is not provided by us.

The following paragraphs provide some further details regarding the layout of the programme and further steps we took to improve its performance. Finally, we present an estimate of the latency introduced by the BoHo.

[1]We chose the name in reference to central access to the subway in the city of Bonn, Germany, called the “Bonner Loch” or “Bonn Hole”, but chose to mix English and German in the name for our programme due to the unfortunate connotations that the phonetics of the pure English version might lead to. Note that a subway station is a perfect metaphor for a wormhole attack since people enter at one end and leave at another station, usually covering a distance that they would not have been able to cover without using a mechanism that is not obvious to the uninformed observer.

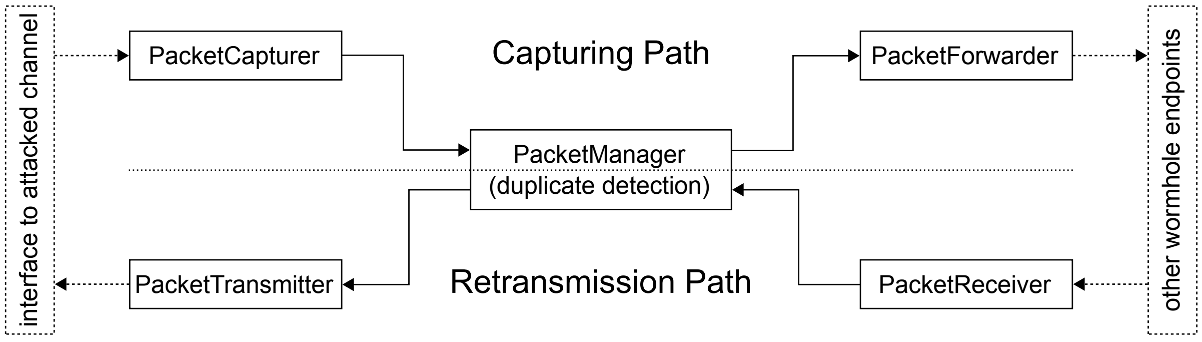


Figure 6.2.: The programme layout of the BoHo wormhole implementation.

Programme Layout

The BoHo is organised in five classes, aided by several supplementaries. These classes evolve around the main tasks of the wormhole:

- Capturing frames from the attacked channel (PacketCapturer)
- Forwarding captured frames to other wormhole endpoints through an out-of-band channel (PacketForwarder)
- Receiving frames from other endpoints (PacketReceiver)
- Transmitting or injecting frames received to the attacked channel (PacketTransmitter)
- Duplicate detection (PacketManager)

Note that we used the ME terminology, i.e. class names refer to packets rather than frames, which would be more accurate since the BoHo captures OSI layer 2 packets.

The classes are organised in two “paths”, the “capturing path” and the “retransmission path”, as indicated in figure 6.2. The former starts with the PacketCapturer (upper left hand side in figure 6.2), which attaches itself to a network device interfacing with the attacked network. Once a frame has been captured, the PacketCapturer calls an instance of the PacketManager to check whether the frame is unique and only if so, it will be handed to the PacketForwarder to be forwarded to all currently connected other wormhole endpoints. The second path starts on the lower right hand side in figure 6.2 and is triggered when the PacketReceiver receives a frame that has been forwarded by another wormhole endpoint. It will call the PacketManager to check for duplicates and schedule non-duplicate packets for retransmission through the network interface attached to the attacked network using the PacketTransmitter.

Our layout comes with several benefits. First of all, it allows us to run an instance of the PacketCapturer and another instance of the PacketReceiver in separate threads to ensure faster processing, particularly when multiple processors or cores are available. Note that however the PacketManager constitutes a bottleneck that cannot be eliminated. Another benefit is that since the paths are independent of each other, they may be deactivated, if the given functionality is not required. While the BoHo is designed to provide a multiple node, bidirectional wormhole, a user may disable one or the other functionality on some endpoints to create unidirectional wormholes. Also, by running two instances of the BoHo on a single node, one of them with the capturing, the other with the forwarding path disabled, it is possible to create a single node wormhole. Note that the duplicate detection step would prevent a single node wormhole from transmitting any frames, if it were to run in a single instance, unless the wormhole had been crafted to bypass it when retransmitting frames captured by itself.

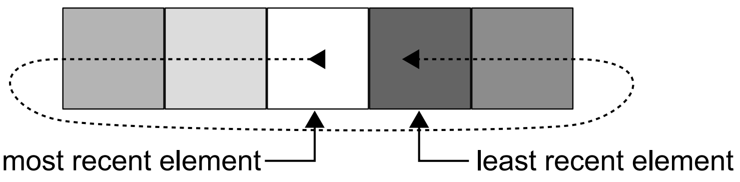


Figure 6.3.: Schematics of the BoHo packet cache. Darker shades indicate a less recently added element, the dashed line indicates the path followed when searching for duplicates, starting with the most recent and ending with the least recent element.

Optimisations

Besides the multi threaded design discussed above, we put some extra care in further implementation details to reduce the delay introduced by the BoHo. First of all, the PacketManager uses a cache of frames either captured by the instance of the programme or received from other wormhole endpoints. The cache includes the full frame, i.e. neither compression nor checksumming are used to reduce the memory requirements. While this may not appear beneficial at first, it allows the PacketManager to trigger the PacketForwarder or PacketTransmitter providing a pointer to its own cache, eliminating the need for extra copies.

The cache discussed in the previous paragraph is maintained in a simple dynamic array or vector which is limited to contain at most a count of elements defined by the user. We do however maintain two offsets indicating the most and least recently added elements in the cache, as indicated by figure 6.3. When checking incoming frames for duplicates, we start with the most recent element and proceed until hitting the eldest, comparing frame lengths and binary matching the frame contents only if the lengths turn out to be equal. A user may thus reduce both the amount of memory and processing power required by reducing the size of the cache, but a very small cache might lead to failure in detecting duplicates.

Cache elements are maintained in instances of the PacketStorage class, i.e. the cache contains pointers to the instances which will be created when the first frames are being added to the cache. Each instance allocates a chunk of memory large enough to enclose not only the frame to store, but also a binary header that would be included when forwarding the frame and a small reserve of unneeded bytes. When reaching the maximum cache size, the PacketStorage instances will be reused, i.e. their contents will be overwritten to include new incoming frames. Whenever the storage's buffer is too small to enclose a new frame, it will have to allocate extra memory. Thus, allocating some bytes in addition to the ones we need when adding a frame may reduce the frequency of memory allocations when frames sizes differ, but not by magnitudes. Note that this scheme implies that BoHo will never free memory allocated for the cache, it allows however to set generous limits for the maximum size of frames given that this maximum is seldomly reached while doing so with an approach which would simply allocate the maximum size for each cache element could quickly turn out to be prohibitively expensive.

Delay Estimate

To determine an estimate of the delay required for a packet to traverse a BoHo wormhole link, we created a simple scenario including four nodes. While we used the ME to provide a virtual network connecting the nodes with each other, we did not use the revised version of the MES described in section but a legacy version that would not model any delays or retransmissions introduced by the wireless link layer, i.e. would deliver frames immediately and only once. We

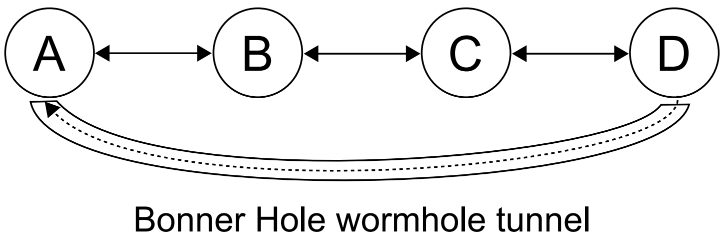


Figure 6.4.: A four node scenario created to determine the delay introduced by the BoHo. Nodes Ⓐ and Ⓓ created a BoHo wormhole tunnel through an out-of-band channel.

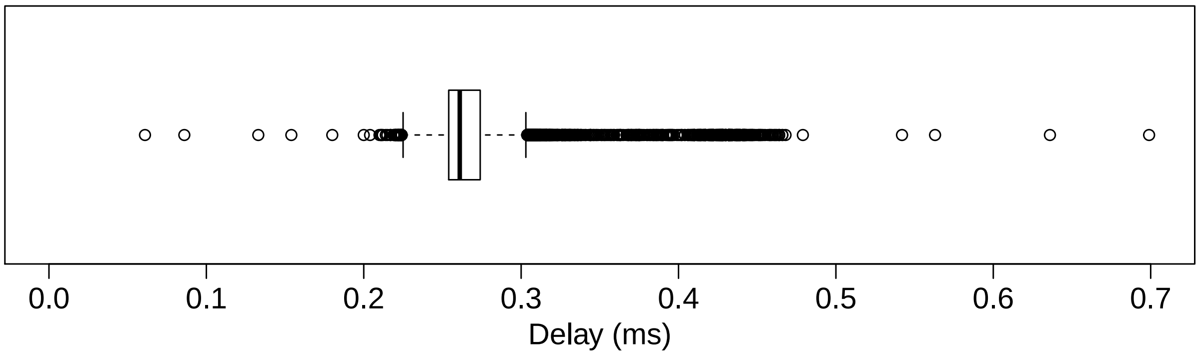


Figure 6.5.: The delay between the direct reception of frames sent from node Ⓒ to node Ⓑ and their retransmission through the wormhole tunnel between nodes Ⓐ and Ⓓ in figure 6.4.

also used a tworay ground model for the wireless channel that would result in frames being relayed to a node if and only if it was within a given range from the sending node. Thus, since all node positions remained static in our scenario, the two-headed arrows connecting the nodes depicted in figure 6.4 directly correspond to the respective wireless connections between the nodes in our scenario. In addition to the given connections, we ran the BoHo on nodes Ⓐ and Ⓓ, creating a wormhole tunnel between them.

We then sent unique packets from node Ⓒ to node Ⓑ, since Ⓓ could overhear the transmission, it would capture them and forward the captured frames to node Ⓐ which would then retransmit them. Since Ⓑ would always receive Ⓒ's transmissions simultaneously with the wormhole endpoint Ⓓ, the first reception of a frame should always be the result of the original transmission while a second reception of an identical frame can only be caused by the wormhole. The difference between the reception of two identical frames thus corresponds to the delay introduced by the wormhole.

Figure 6.5 shows a box plot of the delays obtained from sending unique packets at an interval of 0.15 seconds over approximately 22 minutes. While few frames were received with a delay of less than 0.2 ms, almost three quarters of all frames would be received after at most 0.275 ms. Outliers in figure 6.5 however also tells us that a significant amount frames experienced a delay of more than 0.3 but less than 0.5 ms and none would be delayed by more than 0.7 ms. Of 9520 packets transmitted, a total of 9217 or 96.8% were received twice, i.e. the BoHo failed to deliver 3.2% of the captured frames.

DBrES BoHo attack configuration

Attack time frame	Between seconds 600 and 1000 of the given emulation					
Mark UDP packets to port	32000					
Chosen pairs of attackers	Scenario	0	1	2	3	4
	Nodes (IDs)	{8, 22}	{10, 19}	{10, 12}	{5, 13}	{8, 27}
	Scenario	5	6	7	8	9
	Nodes (IDs)	{2, 30}	{6, 25}	{5, 30}	{10, 19}	{5, 16}
	Scenario	10	11	12	13	14
	Nodes (IDs)	{5, 14}	{5, 15}	{10, 18}	{4, 19}	{20, 23}
	Scenario	15	16	17	18	19
	Nodes (IDs)	{7, 29}	{5, 16}	{14, 29}	{6, 16}	{9, 14}

Table 6.2.: The configuration of the DBrES attack component for the wormhole attack implemented using the BoHo.

Invocation

To execute the wormhole attack in in our emulation environment, we wrote an attack component for DBrES. When set up with a single attacker, it will start two instances of the BoHo listening on different ports an running on the same node, executing a single node wormhole as indicated in Section 6.2.2. The main goal is however to start wormholes involving multiple nodes. To do so, it will start the BoHo executable on each of the given nodes and set them up to connect to each other, but also uses a BoHo feature allowing to set up start and end times so that the attack should start and end simultaneously on all participating nodes. Note that our component does not allow setting up unidirectional wormholes as it would generally be possible with the BoHo programme. While it would be possible to allow configuring that feature through DBrES, we decided to abstain from that in order to maintain a well-readable syntax for the respective configuration.

6.2.3. Configuration in Our Evaluation

Since the wormhole attack is transparent to the attacked network, there are only few options to consider to set it up in our emulations. First, we wanted to execute two attacks in a single emulation but did not want them to influence each other. Thus, the wormhole attack should start 100 seconds after the sinkhole attack's time frame ended at second 500 and last for another 400 seconds. Second, there should be two collaborating nodes executing the attack. We chose those nodes uniformly at random from all nodes except, for the same reasons as those described in Section 6.2.1, the nodes participating in the command and control channel, however when the first nodes had been chosen, we would remove all nodes from its group from the population so that the next node would be in another group.

Reasons for that are obvious, if both nodes participating in a wormhole attack reside in the same location, the attack will merely increase collisions but not provide a significantly more attractive link to the other nodes. Therefore, the wormhole will not attract a considerable share, if any, of the traffic in the network. Since it is rather costly, requiring two nodes and an out-of-band channel, an attacker would probably elect not to do anything, if its only other choice was to execute a wormhole attack with both nodes being placed in the same group.

Besides the aforementioned settings, we had to activate the marking of our simulated user

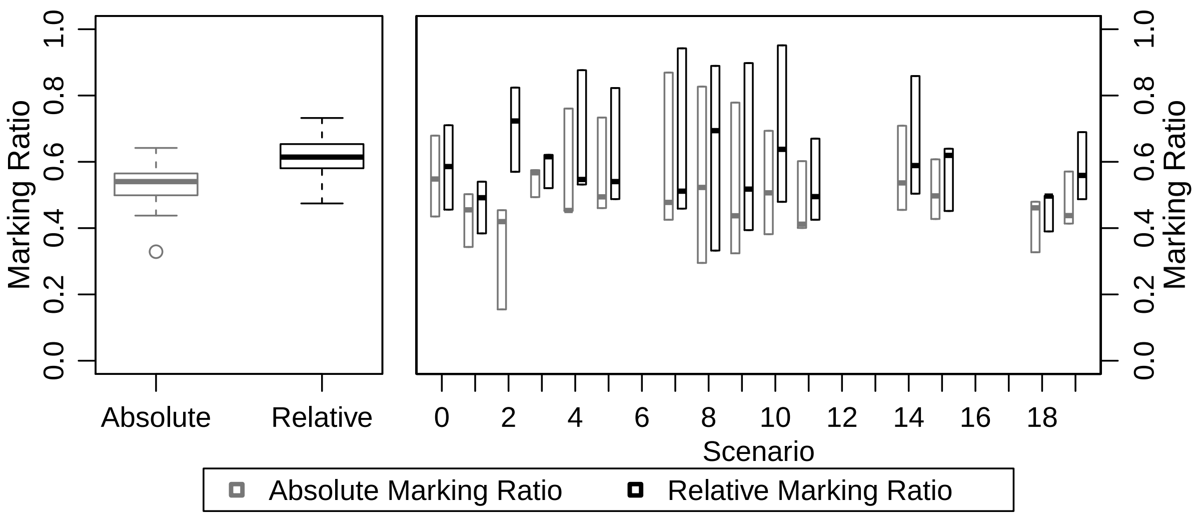


Figure 6.6.: The absolute and relative marking ratios obtained during a wormhole attack. While the left graph shows the totals, the figure on the right hand side shows these values for individual nodes and for each scenario we were able to provide for our evaluation. Note that in this panel, the outer edges of the boxes refer to the extreme values of the respective ratios while the thick line in the box represents the third value.

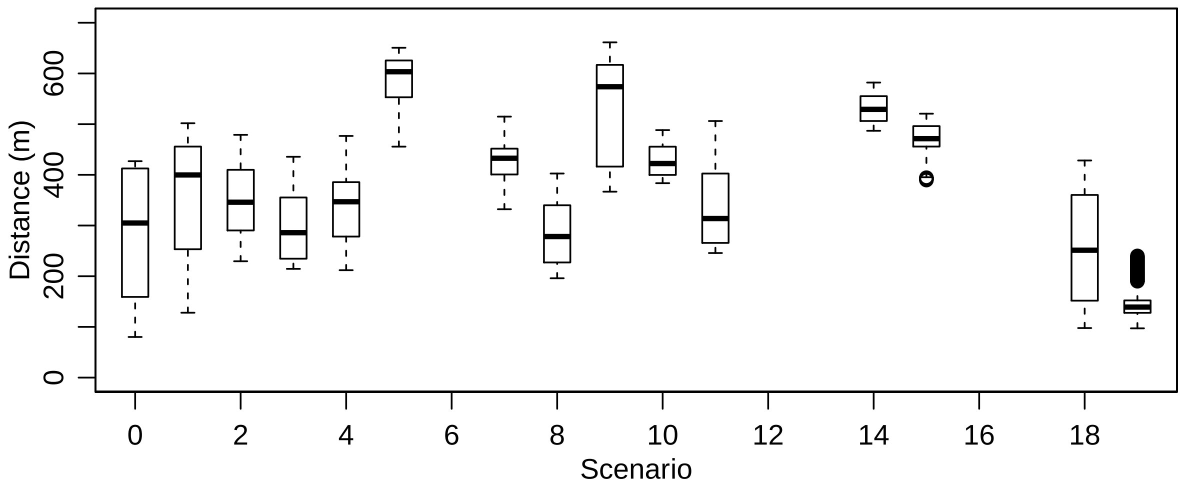


Figure 6.7.: This graph displays the distance between the wormhole endpoints in the attack time frame for each scenario. Distances were determined in steps of 1 second.

traffic, as required by our reasoning in Section 6.2.1. Since the traffic would consist of UDP packets with the destination port set to 32000, we enabled marking only for that specific port. These settings, among with the attackers chosen according to the reasoning in the first paragraph, are reflected in table 6.2.

6.2.4. Results

Figure 6.6 shows the absolute and relative marking ratios obtained in the attack time frame, in total on its left and with details for each scenario on its right hand side. For the latter, each box reflects the highest and lowest per node marking ratio with its upper and lower edge and the remaining ratio with a thick line inside the box. Absolute marking ratios are printed in grey while the boxes reflecting the relative marking ratio defined in Section 6.2.1 have black outlines.

When comparing absolute and relative marking ratios, we see that the former is only slightly lower than the latter, reflecting the generally high PDR in the respective time frames. However, when taking a closer look at the per scenario results, it becomes apparent that the difference may become more significant for individual results. As for the analysis presented in Section 6.1.4, we want to point out that [19] indicates that a MELPe transmission will become inaudible if its PDR drops below 50%. Thus, if the attacker achieved a relative marking ratio of 50%, it has the power to degrade voice transmissions to a degree that the respective node will no longer be able to communicate with the other nodes. On the other hand, while the attackers might overhear some packets that would be routed to their destination without using the wormhole tunnel, it will need an absolute marking ratio of about 50% to ensure that its operators may be able to follow captured MELPe communications.

The obvious implication is that the higher either of the marking ratio, the better for the attacker. Figure 6.6 indicates that the relative marking ratio drops only marginally below 40% for one node and in four scenarios (1, 8, 9 and 18) while remaining above 50% for all nodes in another four scenarios (2, 3, 4 and 14). Surprisingly, at first sight it appears as if there was no direct relation between the distance between the wormhole endpoints, i.e. implicitly the topology of the network, and the marking ratio. Note that for instance Scenario 19 not only exhibits by far the shortest distance between the wormhole endpoints, as indicated by Figure 6.7, but still the lowest relative marking ratio is 48.7%, i.e. just marginally below our goal of 50% for each node.

For the further analysis, we will use the marked graphs of the movement patterns provided in Appendix A.2. They indicate that an important factor concerning the ability of the wormhole endpoints to capture frames and for nodes to receive retransmitted frames is the distance between the respective source and destination and their closest wormhole endpoint. Since the wormhole will neither forge link qualities nor use a stronger radio than the other nodes, there is a chance that routes through legitimate links provide a better aggregated quality than a route through wormhole links. This can be observed in Scenario 11, where one attacked and the non-attacked group overlap for a large part of the attack time frame, resulting in a lower marking ratio for the packets sent by the nodes in these groups.

The contrary effect results in the high marking ratio in Scenario 19. While the wormhole endpoints are close to each other during the attack time frame, they are also particularly close to the member of the command and control group in their own group and thus provide the impression of a very high quality link between these nodes. On the other hand, through being so close to each other, they are not able to provide a benefit for a route to the third node in the command and control group and thus do not achieve significant relative marking ratios on the respective routes.

Another set-up that allows a wormhole to achieve a high marking ratio is when an attacked group is isolated and has no or only low quality links to other groups. In Scenario 10 this results in a marking ratio of 95.1% for the transmissions of Node 11, however while it will also attract traffic directed at that node, it will not be able to tunnel a significant portion of the traffic between the nodes which are not in the isolated group, as indicated by the significantly lower marking ratios for Scenario 11 in Figure 6.6.

Finally, relative marking ratios above 80% and at about 70% for the first and second most affected nodes in Scenarios 2 and 8 appear to contradict the modest distance between wormhole endpoints shown in Figure 6.7. Again, the more detailed data provided in Appendix A.2 provides the clue. During most of the attack time frame, groups are clearly separated and only one of them has two neighbouring groups. This group happens to contain an attacker in both scenarios. Thus, the topology could be described as a chain of groups where the wormhole provides a short cut between two neighbouring links of that chain.

The outer attacker's group will have to communicate to each of the other groups through a relay in the inner attacked group, which becomes cheaper through the wormhole attack. Therefore, the respective command and control node exhibits the highest relative marking ratio. For the other nodes, packets are likely to travel through the wormhole on their way to that node, while relative marking ratios above 50% indicate that at least some of their packets to each other also travelled through the wormhole. Note that in Scenario 8 Node 11 is within an attacked group but moves towards the other other attacked group and away from the wormhole endpoint in its own group, apparently causing the relative marking ratio to reach only 33.2%.

Summarising these results, we find that in our set-up the wormhole would seldomly miss the goal to attract about 50% or more of a node's payload traffic. In total, roughly 60% of the payload traffic that will reach its destination will have travelled through the attacker's wormhole tunnel. While we were able to identify circumstances that appear to be more or less beneficial to the attacker, neither can they be concentrated into a simple recipe nor can the attacker expect to reach full control of all transmissions in the network.

However, this should not be dismissed as a mere linear increase, based on doubling the number of attackers involved, against the expected 30% decrease of the PDR achieved through a sinkhole attack, as described in Section 6.1.4. First of all, the wormhole attack could be carried out without access to any key material. While this would imply that the contents of the traffic would remain obscure to the attacker if it had been encrypted properly, it may still be able to deduce information from it. Methods may include traffic analysis as indicated in [68] but may also be based on knowledge of standards, procedures and situation, e.g. when considering voice communication in military operations. In addition to that, the attacker may decide to build up momentum before dropping particular packets or disrupting the network altogether on moment's notice. This capability renders the wormhole particularly capable of supporting operations in another domain and thus indicates that the wormhole attack may be exceptionally fit for executing CNAs as suggested in [60].

6.3. Summary

This chapter consisted of two main sections. The first section covered the sinkhole attack and started with a description of our metric for determining the attack's impact, a simple comparison between the PDR obtained in the attack time frame, between a replication of the respective scenario with and without an attack. Thereafter, we provided a brief overview to the sinkhole implementation we had written for a prior project and would use for this thesis without major modification. The next section, 6.1.3, described the configuration we used for our evaluation. Since we wanted to maximise the attack's impact, the attacker would claim 20 additional neighbours and maximum link qualities for all existing and forged links. The evaluation results presented in the following section indicate that such a sinkhole attacker may expect to attract roughly 30% of the traffic in our scenarios, however the impact would often not be evenly distributed, regularly allowing the attacker to deprive one node of its ability to participate in communications while not affecting the other's to a significant degree.

The second main section covered the wormhole attack. Again, we started out by introducing our metric for determining the attack's impact. Our "relative marking ratio" metric is the ratio of packets that arrived at their destination and were marked when travelling through the wormhole. Our implementation, the BoHo was written for this thesis and allows, among others, to connect an arbitrary amount of wormhole endpoints with one another. While it also supports single-node and unidirectional wormholes when set up appropriately, we did not cover these variations in our evaluation. Section 6.2.3 describes our settings for the latter. Since the attack itself cannot be parametrised, this was limited to choosing an appropriate pair of nodes as wormhole endpoints for each scenario. We ensured that we would never pick two nodes from the same squad, since these would always be in each other's transmission range and thus offer little, if any, improvement over legitimate transmissions.

In our evaluation, the wormhole attracted more than 60% of the packets that would arrive at their destination in the majority of our scenarios. Thus, the attacker does not achieve full control but acquires the capability to exert significant influence. While we were able to confirm that the topology of the attacked network yields a great influence on the achieved level of control, our evaluation suggests that the distance between the wormhole endpoints will not provide a proper estimate. Last but not least, the pointed out that the wormhole attack provides the attacker with more capabilities when compared against the sinkhole attack and appears thus more appropriate to execute CNA operations as a part of operations in another warfare domain.

Chapter 7

Evaluation of ToGBAD

In this chapter, we describe our evaluation of the ToGBAD approaches introduced in Section 4. While prior evaluations of ToGBAD-LQ and ToGBAD-WH were documented in [10], [11] and [41] respectively, they were based on simulated networks only. An emulative evaluation of the ToGBAD approach was described in [29], however the implementation underwent significant changes afterwards and our implementation of the other approaches was rewritten from scratch. In addition to that, we also introduced the Mean with Node Threshold and Node Picking methods (cf. sections 4.2.3 and 4.2.2) for data aggregation through our implementation of the ToGBAD-LQ concept as a part of this thesis and need to assess their performance in our environment.

This chapter is organised as follows: The first section describes our implementation of the ToGBAD concepts. Section 7.2 then introduces the metrics we will use to rate the success of the ToGBAD detectors. In Section 7.3 we will discuss the parameter space that we will consider in our evaluation and present initial evaluation results obtained with an idealised data set obtained from our emulations. The following section will then discuss the evaluation using the data that was actually available to the detector in our emulations. Finally, we provide a brief summary in Section 7.5.

7.1. Implementation

Our implementation of the ToGBAD concepts is an enhanced version of the ToGBAD component written for the RITA project. While we added some features in the course of this thesis, most of the implementation we use was written by the author on previous occasions, particularly for the RITA project. As indicated by the description of the respective concepts, the implementation is divided into a client and a centralised part where the former runs on a hand-held device to collect data or execute a local detection mechanism and then forward the results to a centralised detector, i.e. the latter. In our set-up, there is an additional middleware component, the Nettalker Server^{NG} (NTS^{NG}), which translates between message formats and aggregates data from various sources, possibly providing it to various consumers.

Since there is only one node running the centralised detector, other nodes must be able to send their reports to that node. This may may require a channel with guaranteed delivery since a node may otherwise not be able to successfully deliver its reports when the network is subject to a routing attack. We also assume that report packets can be dropped by the attacker but that it cannot modify them or assume another node's identity. These properties are easy to achieve using cryptography. More particularly, if the node running ToGBAD shares a unique key with each other node, which may have simply been distributed through appropriate configuration files, these properties can be guaranteed without requiring any asymmetric cryptography operations.

We start out by describing the local components in Section 7.1.1, followed by the NTS^{NG} mediator in Section 7.1.2. The final section discusses the enhanced ToGBAD implementation providing the remote detection in our set-up.

7.1.1. Local Components

The ToGBAD approaches rely on components running on nodes in the networks that would report locally broadcasted and non-forwarded messages including the count of neighbours advertised in them, link quality values that appear to be forged, including the respective magnitude, forged NLQs and finally the positions of the nodes in the network. Since we are using the OLSRd to provide routing in our network, we will need reports on HELLO messages, as well as the ability to determine link qualities and detect their forgery on local nodes and finally we have to determine the positions of nodes in the network.

The RITA Heartbeat programme already generates position reports and sends them to the node that executes our centralised detector. Since the respective information is provided through the RITA MessageEngine at that node, we can retrieve it without requiring any extra messaging. We thus included an interface to the MessageEngine in our reimplementation of the Nettalker Server (NTS), the NTS^{NG}, which will be described in Section 7.1.2.

In the RITA project, the Nettalker Client parsed messages generated by the OLSRd and reported on them along with traffic statistics that were interpreted by the Cluster Based Anomaly Detection (CBAD). However, our implementation of the ToGBAD-LQ concept required an OLSRd plug-in using custom message formats which could no longer be parsed by the Nettalker Client software. Thus, we included the functionality of the Nettalker Client with respect to the OLSRd messages in the OLSRd plug-in described below, where the required information could be gathered at virtually no extra cost.

To implement the ToGBAD-LQ concept, we created an OLSRd plug-in as a part of [12], which also gives some details we will not cover here. It includes a link quality plug-in to the OLSRd, i.e. uses the ETX metric to provide link qualities for all links detected. While the OLSRd employs custom OLSR HELLO and TC message formats for link quality support, the ToGBAD-LQ concept requires the exchange of challenges and responses through HELLO messages, which is not possible with the existing format. We therefore had to introduce our own HELLO messages. These will be transmitted as the payload of regular OLSR message but with the type code set to a custom value of 151. Tables 7.1 and 7.2 show their format. Note that the sender's address, message type, a unique sequence number and the message length are already included in the header of the OLSR message that our message will be encapsulated in.

Our HELLO messages start with a small header that includes the values of the OLSR HELLO message, but also a challenge and the length of both the challenge and the responses included in the payload of the message. Following that, there will be a link message for each neighbour, containing the neighbour's address, link quality values and an arbitrary number, including none, of responses to the challenges received from that neighbour. While our TC messages use the format introduced by the OLSRd for TCs with link qualities, we use different custom type code

Offset	Byte 0	Byte 1	Byte 2	Byte 3
+ 0 bytes	HELLO interval	Willingness	Response length	Challenge length
+ 4 bytes	Challenge (length varies)			

Table 7.1.: Challenge-Response HELLO (IP version 4) message header

Offset	Byte 0	Byte 1	Byte 2	Byte 3
+ 0 bytes	Link code	Link message size	Link quality	Neighbour link quality
+ 4 bytes	Neighbour interface address...			
+ 8 bytes	Sequence number$_0$		Response$_0$...	
+ 12 bytes	...Response$_0$ (length varies)		Sequence number$_1$	
+ 16 bytes	Response$_1$... (length varies)			
...				

Table 7.2.: Challenge-Response HELLO (IP version 4) link message

Offset	Byte 0	Byte 1	Byte 2	Byte 3
+ 0 bytes	Type	Neighbour$_0$...		
+ 4 bytes	...Neighbour$_0$	ETX exceedance$_0$ (float)...		
+ 8 bytes	...ETX$_0$	Neighbour$_1$		
+ 12 bytes	...Neighbour$_1$	ETX$_1$...		
+ 16 bytes	...ETX$_1$	...		

Table 7.3.: Challenge-Response Nettalker Server (IP version 4) report

(152) for them to ensure that nodes will not interoperate with OLSR networks that do not use our mechanism.

The plug-in determines link qualities by monitoring the reception of HELLO messages in a link quality window of size ω. This is done by simply increasing a counter γ_n each time a HELLO message is received from a node n, unless the counter has already reached ω, but also setting a timer to expire after the HELLO emission interval indicated in the message has passed. If that timer expires without the node receiving another HELLO message, γ_n is decreased by 1 and the timer is set again to expire after another HELLO interval duration until γ_n hits zero and n is purged from the nodes neighbour set.

To be able to verify claimed link qualities, each node includes a unique challenge (our method for deriving unpredictable challenges at low cost is described in [12]) in each of its HELLO messages and stores it in a local database. It also stores the link quality values advertised for each neighbour included in the message. When receiving a HELLO message, nodes store the challenge and the sequence number of the message in a database entry referring to its sender. Upon generating its next HELLO message, a node will query that database for each neighbour to check whether it includes any challenges. If it does, the node will calculate the HMAC-SHA1 using its own IP address as a message and the challenge as a shared key. The result will be trimmed to the configured response length and included, along with the sequence number of the message the challenge was retrieved from, as a response in the link message concerning the respective neighbour.

Upon receiving a HELLO message, a node will check for neighbour entries referring to itself. If such an entry exists and contains any responses, it will check for each of them whether it already verified a response from the given node and regarding the given message sequence number. In case the node did not and if the challenge had been generated within the node's own link quality window, it will extract the neighbour's IP address from the OLSR message header and compare the result of calculating and trimming the HMAC-SHA1 in the manner described above with the response provided by the neighbour. If and only if the results match, the node stores that a correct response was received for the respective sequence number, otherwise it stores that the verification failed and will not attempt to verify any further responses from that given node and concerning the given sequence number.

Thus, a node stores the number of HELLOs received from each neighbour and within its own link quality, yielding the quality of the unidirectional link from that neighbour to itself. It also stores the link qualities it transmitted regarding each neighbour, allowing it to verify any claims for link qualities it allegedly reported, and finally the number of correct responses received from each node, again within its own link quality window. Using the latter, we execute the local detection algorithm described in Section 4.2. At first, each node will multiply the unidirectional link quality for a link from a neighbour to itself by the link quality claimed by that neighbour for the reverse link, then it will calculate the sum of correct responses and a value added for tolerance and divide it by its own window's size. If the claimed link quality exceeds the latter value, the node will calculate the normalised exceedance value in accordance with 4.2 and send a report to the centralised detector, using the message format described in table 7.3.

Checking for forged NLQ values is relatively simple. For each NLQ received regarding itself, a node will simply query its local database to check whether it sent the given NLQ value and if so, whether that happened within its own link quality windows. If not, it will send a message to the centralised detector containing the perpetrator's IP address.

To reduce the cost of sending report messages, both the reports generated by the link quality detection mechanism and the HELLO reports introduced at the beginning will be stored in a local database and sent after a short period. HELLO reports are generated at a fixed interval and our NTS^{NG} application allows us to create mixed reports, including report messages of different types, i.e. ideally, link quality reports will be included in the regular HELLO reports. To give users the power to trade-off between lower channel utilisation and faster responses against attacks, users can set a threshold for the age of the eldest entry stored in the database and a report will be sent regardless of any other limits if that age is hit. Finally, a "burst" setting triggers generating reports when too many entries are being stored in the database, this is particularly useful to avoid storing, i.e. delaying, more entries than could be sent in a single packet anyway.

7.1.2. Mediator: The Nettalker ServerNG

The Nettalker Server^{NG} (NTS^{NG}) is an enhanced reimplementation of the NTS written for the RITA project. The latter received reports from the Nettalker Client, translated their binary format into a string representation and provided it to the CBAD and ToGBAD applications, at first only through a text file, later also through a UNIX socket. However, with this thesis came the need to include an interface that would parse position updates provided by the RITA MessageEngine in the XML-based Intrusion Detection Message Exchange Format (IDMEF) (cf. [22]). While our first approach was to include this function in the existing application, this not only further reduced the clearness of the application's code but also would simply not work as it did in the standalone application we had written to test the respective code.

Therefore, we wrote the NTS^{NG} reimplementation. Instead of using the C programming language as for NTS, we use C++, giving us the power to object oriented concepts to provide both a clearer concept and more flexibility. It divides the interfaces into sources and consumers of report entries. A source receives reports through an external interface and translates them into a text format that is understood by detectors relying on the respective report type. Consumers receive the subset of the reports that they require and provide them through another interface to external programmes.

The implementation currently provides two source types, the “IPSource” and the “Msg-SocketSource”. While the former replicates the interface of the NTS, i.e. it listens on a UDP socket, receives and decodes report packets sent by local components, the latter connects to the UNIX socket provided by the RITA MessageEngine and parses the IDMEF messages provided through it but ignores messages other than position reports, which are translated into the WGS84 format (cf. [48]) and then provided to the NTS^{NG} core in a condensed format.

When receiving such parsed reports, the NTS^{NG} core pushes them through to instances of the “SocketConsumer” or “TextfileConsumer” classes. The former provides a UNIX socket that detectors may connect to. Upon connecting, the detector has to request the message types it is interested in, the SocketConsumer will then update an internal list of detector requests and notify the NTS^{NG} core about any changes. This allows the NTS^{NG} core to notify only such consumers about new report entries that actually interested in those reports. In the next step, the SocketConsumer would then consult its internal list to relay the parsed entry to each of the detectors that requested the respective type of reports. As indicated by its name, the TextfileConsumer provides reports through a simple text file. Since there is no channel for requesting or cancelling requests, it always requests any available type of messages, i.e. the text file written by the TextfileConsumer should always contain all parsed reports received by the NTS^{NG}.

7.1.3. The Centralised Detector

The centralised detector is at the core of our efforts. Since it is based on the implementation of the original ToGBAD concept, it still carries its name even though it now implements all three ToGBAD concepts. As indicated in this section’s introduction, it does not interface directly with the client applications described in Section 7.1.1, but either connects to a UNIX socket provided by the NTS or NTS^{NG} programme, reads data provided by them from a text file or can be fed with a stored text file through a pipe for evaluation purposes.

While some parts of the original ToGBAD implementation created for [29] persist, most of the application has been rewritten to achieve better separation of the core functionalities. This was essential to be able to include several types of detectors in a single application. Doing so allows us to run each of those detectors at a time but use the resources required for processing and storing reports just once. Note that it is still possible to run the detectors separately by employing an appropriate configuration for each of several ToGBAD instances.

Figure 7.1 provides an overview to the programme layout resulting from our separation of functionalities. A frontend interfaces with exactly one of the aforementioned data sources and uses incoming reports to updates an instance of the ToGBAD graph object. After an update occurred, it informs an instance of the DetectorHandler class, which refers to an internal data structure to determine which detectors should be informed about the given update and then calls the respective function of those detector instances. While they may not modify the centralised graph object, it provides a backlog of reports, allowing them to reconstruct previous states, if necessary. In addition to using the data provided through the graph, each detector may refer to

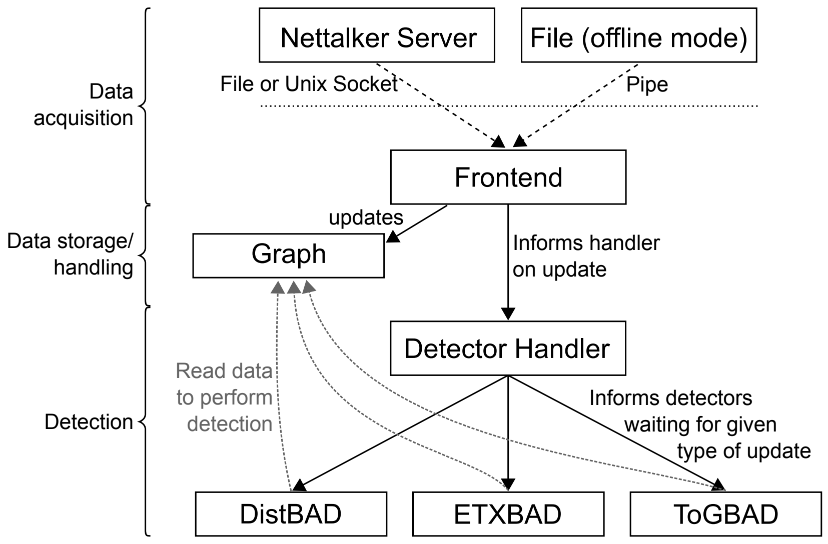


Figure 7.1.: This figure gives an overview on the layout of the ToGBAD implementation. Data acquisition relies on external programmes, i.e. the NTS or being fed through a pipe and uses the acquired data to update a graph object. The object will then be read by the detectors to allow them performing their detection process.

an internal state to render its method possible or speed it up. ToGBAD currently provides three detectors, the design however allows adding other detection methods with few changes to the existing code.

The ToGBAD Graph

The ToGBAD graph is primarily a data structure for storing reports, but also allows retrieving certain properties required for implementing the ToGBAD concepts. It uses an STL map to store references to node objects, allowing quick access to nodes based on their ID. Note that it may not be possible to retrieve properties of the graph which cannot be attributed to a specific node or set of nodes using this data structure. Since our current approaches do not require any such properties, this is however irrelevant at this time.

The node objects mentioned in the previous paragraph are generally created or updated upon receiving a report concerning the given node, some may however be created immediately after ToGBAD's startup. That will be the case if the node has been included in a set of nodes for which the maximum speed and transmission range have been explicitly set up through ToGBAD's configuration file. These properties of a node are required by the ToGBAD-WH approach and thus have to be available for each node. We allow setting them up explicitly through classifying nodes as “vehicle” or “walking” and configuring the assumed maximum speeds and transmission ranges for these classes. Nodes which have not been classified explicitly will fall into the “default” class for which again the given values can be set up through the configuration file.

Other data, namely data received through reports, is attributed with a timestamp of reception or last update. This not only allows detectors to consider the age of an entry in its detection process, but also enables the graph to purge no longer needed data from the graph, keeping ToGBAD's memory footprint stable even when the application runs for a sustained period. To determine the maximum age for entries, each detector is queried for the maximum age of

ToGBAD	ToGBAD detector	ToGBAD-WH	DistBAD
α	alpha	$d_{simple}/d_{advanced}$	algorithm
β	beta	Δ_X	gpserror
ω	factor	ρ	period
learning period	ignoreTime	κ	kappa
ρ	expireIn		threshold
	localAgeing		autoThreshold
			positionValidity
			method

Table 7.4.: The mapping between parameters for the ToGBAD and DistBAD implementations of the ToGBAD and ToGBAD-WH concepts. Note that some parameters used by the implementations are not present in our abstract description.

entries it may be interested in; the graph will then determine the maximum of these ages for all detectors and purge data that exceeds that age periodically.

The DetectorHandler and ToGBAD Detectors

Our DetectorHandler class allows multiplexing incoming data among several detector, even of the same type. If and only if a special flag indicating that ToGBAD is running in evaluation mode is set in ToGBAD's configuration, it will evaluate a variable for each detector that indicates which sections of the file provide settings for that detector. If, on the other hand, the evaluation flag is not set, it will only check whether it is supposed to create a detector for each given type and set it up using the standard configuration section for that type of detector. Once a detector has been set up and read its own configuration, it will be queried for the maximum age of report entries required for its mechanism and the report types that need to be incorporated in the graph as well as those it should be triggered on. Thus, the DetectorHandler does not need any information on the internal workings of a detector as long as it implements the required interfaces.

In addition to services described above, a recent addition lets the DetectorHandler provide a notification service to the detectors. If a detector detects an ongoing attack, it will notify the DetectorHandler which in turn notifies other RITA components through sending an SNMP trap and stores the event in a log file.

At this point, we want to briefly introduce the implementation of the ToGBAD concepts. The original ToGBAD concept is implemented by the ToGBAD detector, which is a modified version of the implementation described in [29]. We latter added the Distance Based Anomaly Detector (DistBAD) implementation of ToGBAD-WH and ETX Based Anomaly Detector (ETXBAD), implementing ToGBAD-LQ. While we were aware of previous implementations of ToGBAD-WH had been developed for [11], it lacked structure and thus we decided to reimplement the concept from scratch for the RITA project. Our implementation was later extended as a part of [41]. ETXBAD was written in the course of [12], as for ToGBAD-WH, a previous implementation, written for [10] existed, but since it was supposed to provide nothing but a basic implementation of the concept, it consisted of a Perl script that only provided a background for writing the respective detector as a part of the ToGBAD application.

ToGBAD While the ToGBAD algorithm described in Section 4.1 only considers a globally aged mean and deviation, our implementation provides a method for determining these values

Mean		**Mean with Node Threshold**		**Node Picking**	
	method = 0		method = 1		method = 2
χ	threshold	χ	threshold	π	filter
π	filter	π	filter	ρ	period
ρ	period	ρ	period	η	minReporters
		η	minReporters		

Table 7.5.: ToGBAD-LQ parameters and their equivalent in the ETXBAD implementation. Since we described three approaches for ToGBAD-LQ, each with its own configuration, we split the relevant parameters accordingly. For each approach, the parameters used in Section 4.2 are displayed on the left while their equivalents in our implementation are shown on the right hand side of the respective table.

mean for each node individually. [23] however pointed out that using the local mean may not be beneficial and thus we will not use this feature in our evaluation. Table 7.4 provides an overview to the mapping between the configuration settings of the ToGBAD implementation and the description provided in Section 4.1 on its left hand side. This includes the localAgeing parameter which toggles the aforementioned behaviour which is not described there. Note that the implementation provides additional parameters which are however not related to the detection process and have thus been omitted in the table.

DistBAD Our implementation of the ToGBAD-WH centralised detector interprets the reports on received HELLO messages by the ToGBAD detector as connectivity reports. While this could be exploited by a wormhole attacker that would simply not forward any frames containing HELLO messages, doing so would result in the nodes receiving tunnelled packets not detecting the neighbourship relations created through the wormhole attack. Therefore, they would determine their routes as if the connections added by the wormhole attack did not exists, i.e. the attack would not alter any routes and could thus not be considered a successful routing attack. This has the nice effect that, in contrast to the implementation described in [11], DistBAD does not require any additional messaging if both the RITA Heartbeat and ToGBAD detector are being employed in a set-up anyway.

The right hand side of table 7.4 shows the mapping between the parameters of the DistBAD configuration and the respective symbols used in Section 4.3. Our implementation still provides the link based detection method implemented but discovered to perform poorly in [11] and thus employs the option "method" to select between this and the more advanced period based approach. "autoThreshold" allows choosing between the original behaviour of that approach, requiring a fixed threshold chosen through the parameter with that name, and the adaptive method developed in [41]. Finally, "positionValidity" indicates the maximum age of a position in seconds until it will no longer be considered in the detection process. This is vital when using the "algorithm" parameter to select the advanced method for determining the distance between two nodes, since with increasing age of position updates, nodes may be farther away from each other without triggering an alarm for the distance of frames travelling between them. As for ToGBAD, our table does not display parameters that are not related to the detection process.

ETXBAD Last but not least, ETXBAD implements the central detector of the ToGBAD-LQ concept. Since the method described in Section 4.2.1 did not perform well in the evaluation

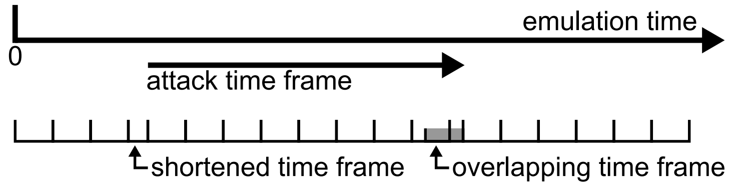


Figure 7.2.: For determining false positive/negative ratios, the emulation time will be divided in time frames, indicated by small brackets below the bar representing the emulation time. Time frames will be shortened when an attack starts within them or moved to align their end with an attack's end, if it would otherwise occur within the given frame.

of ETXBAD presented in [12], we suggested two additional methods in sections 4.2.2 and 4.2.3. Each of these methods comes with its own set of parameters which we thus present in different subtables of table 7.5. Note that when setting the "minReporters" parameter to 1 our "Mean with Node Threshold" method becomes equivalent to the "Mean" method. The "method" parameter chooses among the named methods where the value required to select each method is indicated in the respective subtable. Note that the interpretation of η or minReporters changes depending on which method is selected (cf. sections 4.2.2 and 4.2.3). As for the other approaches, we omitted parameters that are not related to the detection process in table 7.5.

7.2. Metrics

We want to define metrics that are applicable to each of the three detectors that compose the ToGBAD programme. Also, the metrics should allow comparing this evaluation's results against that of other approaches, i.e. be as general as possible. The obvious choice is thus to observe the ratios of false positives and negatives, the most common metric for intrusion detection systems. In addition to that, we want to determine the delay between the begin of an attack and the first genuine alarm produced by ToGBAD. While the former metrics indicate the level of trust a user should generally have into an alarm generated by ToGBAD, the latter indicates whether alarms occur in a timely fashion, where a reaction could prevent further damage from materialising, or not.

Since each detector may produce a series of alarms both during an attack or a phase without any attack, we have to provide a means of normalising the base for determining detection ratios. We do so by dividing each emulation's time space in sections or time frames of 5 seconds each. If however, a given type of attack starts within a time frame, it will be shortened so that the next time frame's start will align with that of the attack. If, on the other hand, an attack would end within a time frame, the section will not be shortened but its start will adjusted to align its end with that of the attack. Figure 7.2 illustrates this approach; a time bar represents the emulation time, an arrow below that bar an attack taking place during the marked time frame. Small brackets below the time bar indicate the time frames created using the described method, they are even lengthed until a frame would overlap with the attack's start and is thus shortened, another section (marked with a grey background) is adjusted to align its end with the end of the given attack and thus overlaps with its proceeding section. We describe further details for determining the given metrics for each of the detectors in the sections below.

We also define the notion of a failure to detect in regard to the delay metric. An attack should only be considered to have been detected, if the respective alarm is generated considering data that may have actually been caused by the given attack. Thus, if an alarm occurs that appears to refer to the attack but occurs at a point in time where not only the attack ended but, when considering the maximum report delay and age of data considered in the detection process, also the decision cannot be based on data generated during the attack time frame, we consider the detector to have failed to detect the attack. In some cases, as described below, there may be several effective attack time frames; in these cases, we will consider the mean values of the delays where the attack has been detected successfully as representing the detection delay for the given scenario while failures to detect will be weighted by dividing their count through the total count of delay values, both successful and failed. Since a reaction to the first attack may serve to avert successive attacks, we provide the given values separately only considering the first attack time frame where several time frames are possible.

7.2.1. Topology Graph Based Anomaly Detector

ToGBAD detects nodes that forge neighbourships in OLSR networks. Since this by definition excludes neighbourships that were determined through the OLSR link sensing mechanism, an attack occurs if and only if a node that has been set up as an attacker using the mechanisms described in Section 6.1.2 emits messages that contain neighbourship advertisements that have not been determined through the OLSR link sensing mechanism. We will call such messages "forged messages" even though they may contain genuine entries. Note that, depending on the set-up, a node may be set up to forge neighbourships at a given point of time but emit messages without any forged neighbours, since all the neighbours it was set up to forge are in fact genuine neighbours; our definition implies that those messages will not be forged and an attack is not taking place while that is the case.

Considering the definition of an attack described in the previous paragraph, we define an attack to be starting with the first forged message and ending with the last message emitted before a message that does not contain any forged neighbours. A true positive thus occurs when an alarm is triggered for the attacking node and in between those messages. Since this definition discriminates against alarms that refer to the attacking node but are created in reaction to and thus after the emission of the last forged message, we consider alarms that are created in the 5 seconds after its emission to be true positives. Generally, true positives will be attributed to the time frame they occur in, in the special case of an alarm occurring after the last message of an attack time frame, we attribute it to the last frame within the given attack.

To calculate the ratios of false positives and negatives for a single time frame t, we consider the size of the sets of nodes α_t, for which an alarm was triggered within t, β_t of nodes executing an attack in the given time frame and γ_t that do not. Thus, we can define the ratios of false positives and negatives as below:

$$\begin{aligned}\text{Ratio of false positives}_t &= \frac{|\alpha_t \setminus \beta_t|}{|\gamma_t|} \\ \text{Ratio of false negatives}_t &= \frac{|\beta_t \setminus \alpha_t|}{|\beta_t|}\end{aligned}$$

Note that the ratio of false negatives is not defined when there are no attackers in t, which however perfectly matches our intuition of a false negative ratio.

To determine the aggregated ratios, we determine the ratios among a complete scenario as the arithmetic mean of the individual ratios. However, we have to consider that while ToGBAD is not designed to detect wormhole attacks, nodes overhearing both the original transmission and a retransmission of a HELLO message would report the combined count of neighbour entries to the central detector. Since this would double the count of neighbours reported for a node, it would most likely be identified as a sinkhole attacker while the cause would have been a wormhole attack perpetrated by another node. To avoid having to classify these either as false positives, since the reported node did not perpetrate an attack, or true positives, as the alarm is caused by an actual ongoing attack related to the reported node, we decided to exclude the time frame during which the wormhole attack takes place, including the first time frame after the wormhole attack stopped, from our evaluation of the ToGBAD detector. Thus, if T is the set of all time frames in the scenario, A is the set of all time frames in which a sinkhole attack occurs and W_{+1} is the set of time frames concerning the wormhole attack as described above, this leaves us with:

$$\begin{aligned}
\text{Ratio of false positives} &= \frac{1}{|T \setminus W_{+1}|} \cdot \sum_{t \in T \setminus W_{+1}} \frac{|\alpha_t \setminus \beta_t|}{|\gamma_t|} \\
\text{Ratio of false negatives} &= \frac{1}{|A|} \cdot \sum_{a \in A} \frac{|\beta_a \setminus \alpha_a|}{|\beta_a|} \\
&= \sum_{a \in A} \frac{|\beta_a \setminus \alpha_a|}{|A| \cdot |\beta_a|}
\end{aligned}$$

Defining the delay of the first genuine alarm for an attack is simple. We define it as the time that passes between the transmission of the first message in an attack time frame and the first alarm that occurs after that message and refers to the node that generated the message. Since our definition of an attack time frame may lead to several attack time frames occurring in a single emulation, even if only a single attack has been set up, we define the scenario's delay as the arithmetic mean of all individual delays, as described in Section 7.2.

7.2.2. ETX Based Anomaly Detector

The ETXBAD detects forged link qualities in MANETs using a modified OLSR protocol with link quality extensions. To define false positives and negatives in this context, we define forged messages as messages that contain at least one link quality entry that exceeds the link quality determined through the regular link sensing mechanism. Note that this definition excludes attacks where the attacker propagates lower link quality values. While an attacker may modify routes by doing so, it will generally divert routes away from itself, i.e. reduce its control over the traffic in the attacked network rather than enhancing it. Since that would not be in line with the goals we defined for sinkhole attackers in Section 6.1, we restrict our notion of an attack accordingly.

Given the definition of forged messages provided in the previous paragraph, we may apply the definitions of false positives, negatives and delay given in the previous section correspondingly. While the potential for the wormhole attack confusing the detector does not exists here, we also exclude the respective time frames here, to maintain a consistent population for our evaluation.

Thus, if α_t is the set of nodes for which an alarm was triggered, β_t the set of nodes executing an attack and γ_t the set of nodes not executing an attack in time frame t, when referring to a set T of all and A of time frames in which an attack occurs and finally a set W_{+1} of wormhole attack time frames as in the previous section, we define the ratio of false positives as below:

$$\begin{aligned}\text{Ratio of false positives} &= \frac{1}{|T \setminus W_{+1}|} \cdot \sum_{t \in T \setminus W_{+1}} \frac{|\alpha_t \setminus \beta_t|}{|\gamma_t|} \\ \text{Ratio of false negatives} &= \sum_{a \in A} \frac{|\beta_a \setminus \alpha_a|}{|A| \cdot |\beta_a|}\end{aligned}$$

The delay of the first genuine alarm will again be the time that passes between the transmission of the first forged message in an attack time frame and the first alarm referring to the node that generated the message. As in the previous section, we will aggregate any delays obtained within a given emulation by calculating their arithmetic mean.

7.2.3. Distance Based Anomaly Detector

DistBAD detects wormhole attacks against MANETs. Since these are generally completely transparent to the attacked network, we cannot pinpoint the start of the attack on the transmission of a certain kind of messages as we did for ToGBAD and ETXBAD. We thus define an attack time frame to start when at least two wormhole endpoints indicated that they are ready to capture and retransmit frames and assume an attack to be over when all but one wormhole endpoint indicated that they are shutting down and will no longer capture or retransmit any frames. Note that the endpoints may still require some time to connect to each other or transmit frames queued for retransmission before being able to start or completely cease operation, the actual begin and end of the attack may thus differ by fractions of a second from the attack time frame as we define it.

Since DistBAD does not identify possible attackers but merely indicates that an attack appears to be ongoing on a given set of links, we define a true positive as an alarm which is triggered within an attack time frame. As for the other detectors, we consider alarms that are triggered within the 5 seconds following the end of an attack time frame to be true positives, but attribute them to the proceeding time frame.

While DistBAD creates an alarm when encountering outdated positions in its detection process, this method was not part of our considerations for this evaluation and such alarms will thus be classified as true negatives. Note however that this mechanism would be triggered by a successful sinkhole attack since it would suppress position reports, preventing DistBAD from executing its detection process for the affected nodes. Thus, in line with the arguments given in Section 7.2.1, we exclude the sinkhole attack time frames, including the time frame following the last attack time frame, from our evaluation of the DistBAD detector.

With sets T containing all time frames, A containing the time frames in which an attack was ongoing, S_{+1} of sinkhole attack time frames including the following time frame and finally B of time frames in which an alarm was triggered, we define:

$$\begin{aligned}\text{Ratio of false positives} &= \frac{|B \setminus S_{+1} \setminus A|}{|T \setminus S_{+1}|} \\ \text{Ratio of false negatives} &= \frac{|A \setminus B|}{|T \setminus S_{+1}|}\end{aligned}$$

The first genuine alarm thus refers to the first alarm triggered after the start of an attack time frame and the time in between constitutes the respective delay. Since with this definition several attack time frames will only occur if set up specifically, we do not define a method for aggregating delays at this time.

7.3. Parameter Selection

Since most of the parameters for the ToGBAD detectors have an infinite space of potential settings, we had to choose a reasonable subset of the possible configurations as a base for our evaluation. For obvious reasons, we would exclude such settings that had shown weaker performance in earlier evaluations, except for ETXBAD where we want to compare the performance of the existing and our suggested aggregation methods.

In addition to choosing the parameter space, we want to determine a set of parameters that works well under ideal conditions, i.e. when reports would be received without delay and would never get lost. This allows us to assess the impact of a lossy channel on the ability of the ToGBAD detectors to detect an ongoing attack. Thus, we generated report log files from data collected locally at each node to provide data that would neither be affected by any loss of connectivity nor subject to the attacks. In addition, we extracted the positions for each node from the data we would provide to the MES and wrote their position to the respective data file in steps of one second.

We start out by briefly describing the chosen parameters for the local component in the next section. Following that, we describe the parameter set for our evaluation of the ToGBAD detector and present results obtained with the idealised data obtained in the manner described in the previous paragraph in Section 7.3.2. In the following sections 7.3.3 and 7.3.4, we do the same for the ETXBAD and DistBAD, respectively.

7.3.1. Local Detection

The local detection is a part of an OLSRd plug-in described in Section 7.1.1. Other than the centralised detector, it would directly interact with other components in the network, preventing us from using idealised data or, given the challenges described in Section 5.3, evaluating different parameter sets. We thus had to settle for a single plausible configuration.

We set the HELLO and TC interval to 2 or 5 seconds respectively, representing the respective standard values for the OLSRd. While we had used a link quality window size of 10 for the evaluations presented in [12], this is often described as the lowest sane value for a link quality window. Thus, we chose the next higher value that would divide 255 since this would allow our plug-in to encode link quality values, i.e. the fraction of HELLO messages received within window size, without loss of precision in a single byte. As in [12], we turned the tolerance

Local detection settings	
HELLO interval	2 seconds
TC interval	5 seconds
windowSize	15
tolerance	0
helloReportInterval	5 seconds
notificationDelay	5 seconds
notificationBurstThreshold	170 reports

Table 7.6.: This table reflects the settings for the OLSRd plug-in that constitutes the local component of our detectors. Since we could not modify these settings without running an extensive amount of additional emulations to obtain reliable results, we did not evaluate alternatives for these settings.

parameter off, i.e. did not accept any deviation at all. Note that we do not recommend this for any purposes other than evaluations.

Setting the HELLO report interval to 5 seconds basically replicates the behaviour of the predecessor, i.e. Nettalker Client, in previous evaluations, where it would send a report every 5 seconds. However, our plug-in also allows for setting up a delay for sending reports on forged link qualities, the notification delay, which we set to the same value. Finally, by setting up a burst threshold of 170, we trigger the burst mechanism described in Section 7.1.1 when the stored count of reports approaches the capacity of a single report packet. Table 7.6 provides these values for your reference.

While not strictly a part of our local detection components, the RITA Heartbeat component provided positioning information in our emulations that would be extracted by the NTS^{NG}. The nodes would generated the respective reports at an interval of 5 seconds.

7.3.2. Topology Graph Based Anomaly Detector

The ToGBAD detector uses several parameters that directly or indirectly affect the detection process. The "expireIn" setting determines the maximum age of reports that would be considered when determining whether a link exists between two nodes. A node will report another node as a neighbour while at least one link quality value is above 0 and only the link quality will deteriorate when a node no longer receives HELLO messages from another node. Since the nodes are set up with a link quality window of size 15 and emit a HELLO message every two seconds, the maximum duration a node will report another node as a neighbour without receiving HELLO messages from it is 30 seconds and thus we set up "expireIn" accordingly. Since nodes are supposed to send reports every 5 seconds, setting "ignoreTime" to 10 seconds implies that we should have received at least one but up to two reports from each node when ToGBAD starts its detection process.

While the aforementioned parameters affect ToGBAD's detection process indirectly, the "alpha", "beta" and "factor" settings are directly related to it. Since the former two parameters both control the ageing for variables related to each other, they should always be set to the same value. In [27], the performance of ToGBAD decreased for values larger than 0.05, so we will evaluate values 0.01, 0.03 and finally 0.05 but no values larger than that. We will set the factor to 1, which performed best in [27], 3 and 5. Table 7.7 provides an overview to these values for your reference.

Figures 7.3 and 7.4 show the results of running ToGBAD with the described parameter sets

ToGBAD detector evaluated parameter sets	
alpha = beta	$\{0.01, 0.03, 0.05\}$
factor	$\{1, 3, 5\}$
ignoreTime	10 seconds
expireIn	30 seconds
localAgeing	false

Table 7.7.: This table reflects the parameter space we explored for our evaluation of ToGBAD. Where values are given as a list, we evaluated each possible combination, while the ignoreTime, expireIn and localAgeing settings were set up identically in each parameter set.

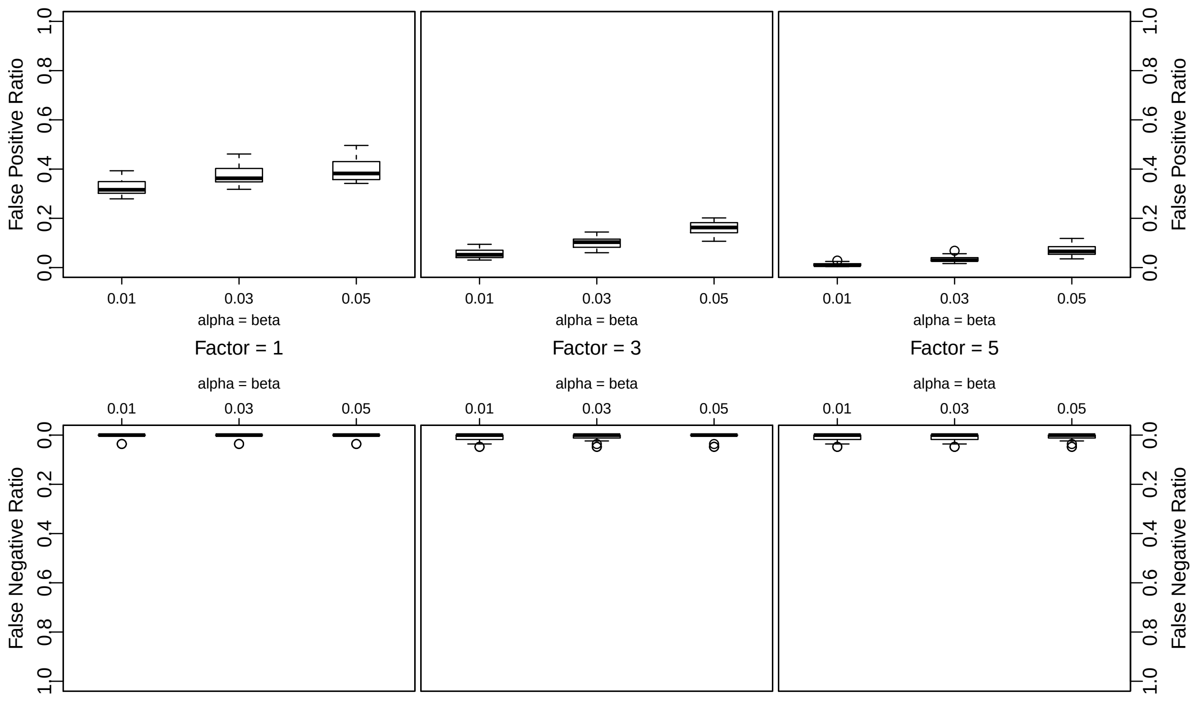


Figure 7.3.: False positives and false negatives obtained when evaluating the ToGBAD detector with idealised data.

on an idealised data set. While a factor of 1 yields almost no false negatives for any setting of alpha and beta, it also results in a significant ratio of false positives, not achieving a ratio of less than 25% in any scenario. While results for factors 3 and 5 exhibit a slightly larger ratio of false negatives, the increased factor yields a significantly lower false positive ratio. Chances are that with the given idealised data set, increasing the factor setting would yield an even more favourable false positive ratio. However, a larger factor would also facilitate attacks where the attacker tries to trade-off between the potential impact of its attack and its detectability and thus we do not regard factors larger than 5.

Focusing on the alpha and beta settings, we observe that lower values appear to decrease the false positive ratios but also promote a slight increase in the false negative ratio. However the median for the false positive ratios remains at 0 for each of the parameter sets displayed. Since we handle, as pointed out in Section 4.1, negative differences between a node's count of neighbourships in the graph and the count of neighbourships it claimed as if they were 0,

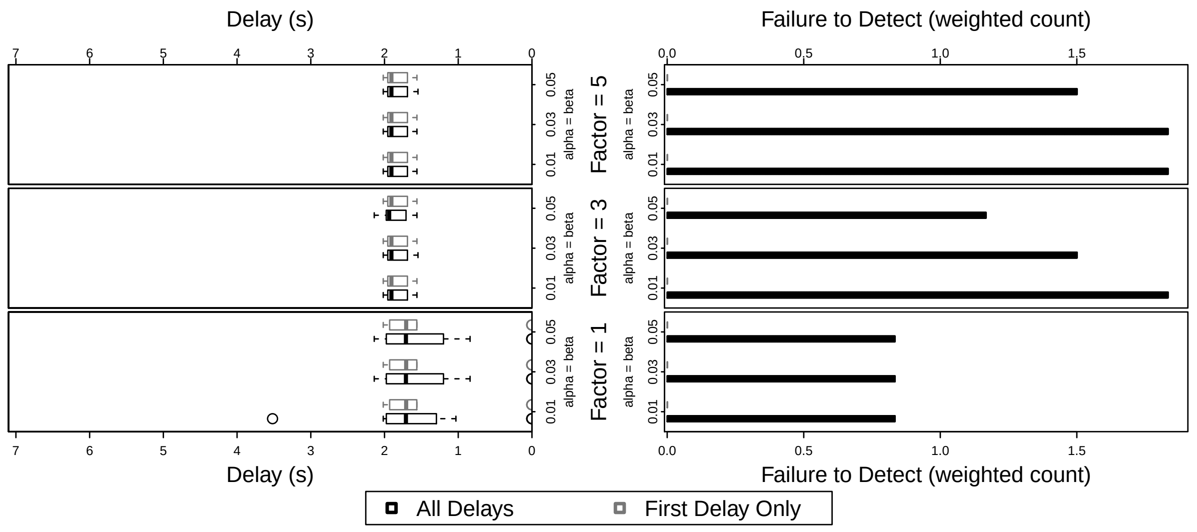


Figure 7.4.: Delays obtained when evaluating the ToGBAD detector with idealised data. Note that the x-axis is into to halves where the half on the left hand size uses normal scaling while the axis on the right hand side is scaled logarithmically. Black boxes represent the delay determined as described in Section 7.2.1 while grey boxes are based on the first delay for each scenario only.

lower alpha and beta values not only imply that the aged mean and deviation will be adjusted to higher values more slowly, but also the same for their downward adjustment. Therefore, once adjusted upwards, the detector will be more likely to accept larger difference values for both malicious and non-malicious nodes.

Figure 7.4 shows the delay between the first emission of the first forged HELLO message for both the first attack time frame only and for all attack time frames, as defined in Section 7.2.1. Since reports will not be delayed, the maximum acceptable delay depends only on the period of 5 seconds and the duration between the first and last HELLO in an attack time frame. All but the first attack time frame contained just two HELLOs, thus the maximum acceptable detection delay for these is 7 seconds. As indicated by the figure, ToGBAD fails to detect some of these later attack time frames for each configuration, increasingly so with an increased factor, but never fails to detect the first attack time frame.

For each pair of configurations and scenarios the delay for detecting the first attack is almost exactly between 1.5 and 2 seconds, except for one scenario in which settings with the factor set to 1 appears to result in instant detection. This may however be linked to the high ratio of false positives pointed out above, where chances for a non-malicious node triggering an alarm are already between about 25% and 50%. Also, a lower factor supports faster detection of additional attack time frames while configurations with the factor set to 5 appear to fail completely at detecting them, as indicated by the identical boxes for representing the detection delay for both first and all delays.

Nevertheless, configurations with a factor of 5 exhibit a promising ratio of both false positives and false negatives while there is no significantly longer delay for detecting the first attack. Given the very minor increase in false negatives but relatively strong decrease in false positives, the factor should be combined with the lowest value for alpha and beta, i.e. 0.01. While these configurations do exhibit a weakness in failing to detect the described short additional attack time frames, it remains questionable whether an attacker would actually be able to attract any

Mean		**Mean with Node Threshold**		**Node Picking**	
method	0	method	1	method	2
threshold	$\{0.1, 0.3, 0.5, 0.7\}$	threshold	$\{0.1, 0.3, 0.5, 0.7\}$	filter	$\{0.1, 0.3, 0.5, 0.7\}$
filter	0	filter	0	period	5 seconds
period	5 seconds	period	5 seconds	minReporters	$\{3, 5, 10\}$
		minReporters	$\{2, 5\}$		

Table 7.8.: The explored parameter space for the evaluation of each of the ETXBAD approaches. Note that while we disable filtering for the mean based methods, we evaluate settings for Node Picking where it is turned on. We do that to take into consideration that Node Picking does not by itself distinguish between reports indicating minor or significant exceedance values.

routes with such an attack and thus we consider this weakness negligible when considering the benefit of the improved false positive and negative ratios.

7.3.3. ETX Based Anomaly Detector

Since we are using three different methods for aggregating data through ETXBAD, we have to define parameter sets for each of them. However, the period parameter is shared by each of the methods and with identical meaning. It will be set to 5 for each of the approaches. This is motivated by the 5 second report interval used by the local detection mechanism, i.e. it gives each node a fair chance of contributing to the detection process. We will describe the parameters used by the Mean and Mean with Node Threshold methods in the next paragraph and the parameters for the Node Picking approach in following one; we also provide the selected parameters in table 7.8.

The “threshold” setting defines the lowest mean value that would trigger an alarm. While reports may include values ranging from 0 to 1, where 0 would indicate that no attack is taking place and must thus be excluded from the set, on the other hand, a value of 1.0 is almost impossible to achieve, since this would imply that none of the reporting nodes received a valid response within the last 30 seconds before generating their reports. While this could easily occur for a given node, it is very unlikely to occur for all nodes in question. Thus, we start at a value of 0.1 and increase it in steps of 0.2 until reaching 0.7. Note that the mechanism controlled by the “filter” setting is targeted at preventing attacks that try to lower the aggregated mean value. Since this attack will not be perpetrated in our evaluation scenarios, we set this to a static value of 0.

Finally, we will set the “minReporters” setting for the Mean with Node Threshold’ approach to 2 and 5, where 2 constitutes the lowest values distinguishing the approach from the regular Mean approach and 5 already corresponds to half a group reporting on forged link qualities for a suspected attacker where nodes within the attacker’s group will usually have high quality links to the attacking node, limiting the number of nodes for which the attacker can forge link qualities in the first place, at least when not combining link quality forgery with neighbourship forgery.

Our Node Picking approach employs only the ”minReporters“ to directly influence the detection process. While the argument given for the minReporters parameter supplied by the Mean with Node Threshold method applies here as well, for this method it considers two types of reports where each node may contribute reports of each type. Thus, the maximum value will be twice the value given above or 10. The lowest value should be 3, as described in Section

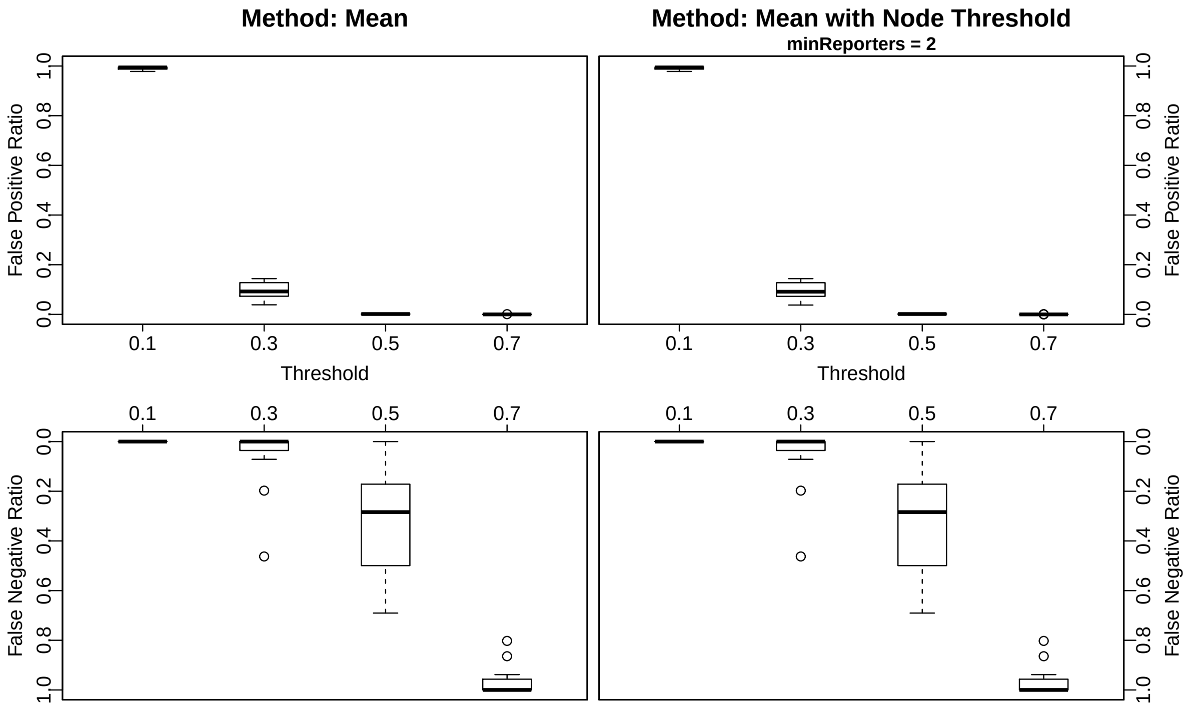


Figure 7.5.: Evaluation results for the ETXBAD detector using the Mean and Mean with Node Threshold methods and idealised data.

4.2.3, since it implies that at least two nodes must contribute reports when we assume an attack is ongoing. This would effectively prevent any attack where the attacker tries to accuse other nodes of perpetrating an attack, unless there are at least two cooperating attackers. To be able to predict the effect of increasing the value for ”minReporters“, we include 5 as a third value.

While the ”filter“ setting shares the same name and fundamental functionality in the implementation, the task of preventing accusation attacks is performed by the ”minReporters“ for this approach. On the other hand, the ”filter“ serves as an implicit threshold that determines the minimum value a node has to report in order to be considered victim of a link quality forgery attack. Thus, we do not set it to a static value as for the mean based approaches, but use the same parameter space we used for the ”threshold“ setting there.

Figure 7.5 shows the ratios of false positives and negatives for the existing ”Mean“ and the ”Mean with Node Threshold“ approach introduced in this thesis, where for the latter ”minReporters“ was set to 2. It is apparent that the results for the latter method, shown on the right hand side of Figure 7.5, are identical to the ones depicted for the former method on the left hand side of the figure. Since the same is true for the results with ”minReporters“ set to 5, we abstain from discussing them here, but provide them in Appendix C.1.

The results show a dilemma we already pointed out in [12]; while it is possible to achieve a false positive ratio of almost 0, namely with thresholds 0.5 and 0.7, the ratio of false negatives will increase drastically when choosing either of these values. With the next lower threshold, 0.3, there are no false negatives in more than half the evaluated scenarios, but also one half of the scenarios exhibit a false positive ratio of about 10% or more. Only with a threshold of 0.1 no false negatives will occur, however at this value, there will be almost no true negatives either, thus it appears impossible to provide a sufficiently general configuration which does not suffer from an undesired rate of either false positive or negatives even when using idealised

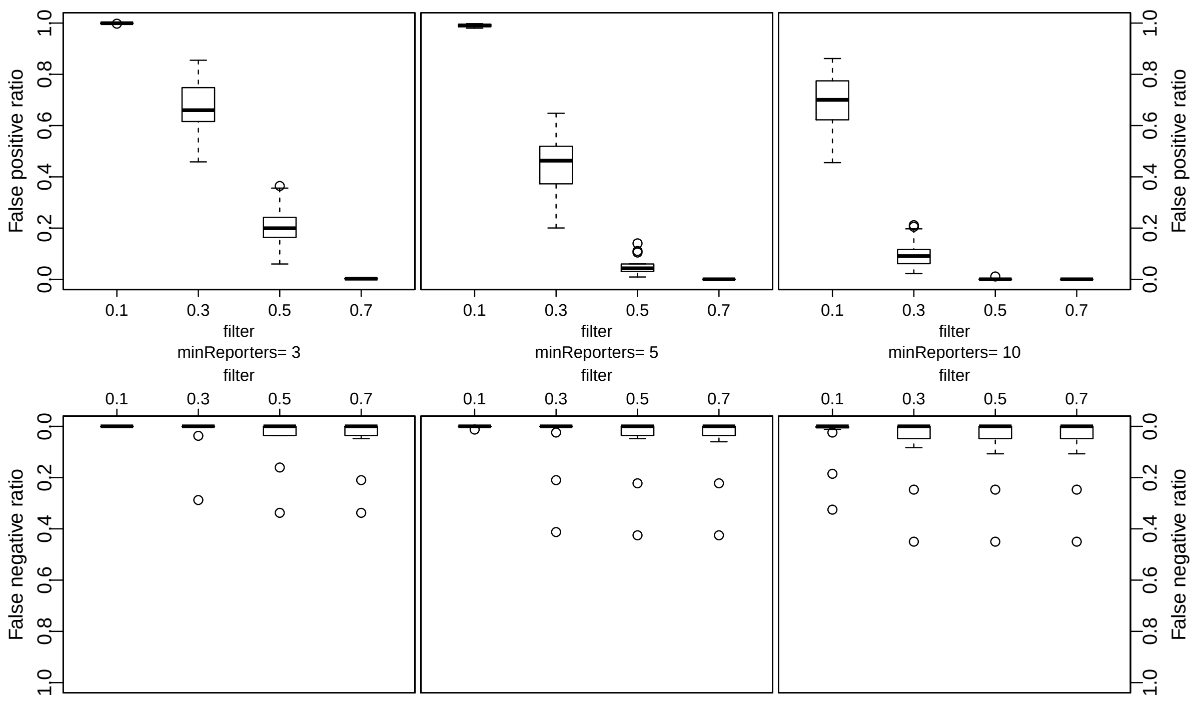


Figure 7.6.: Evaluation results for the ETXBAD detector using the Node Picking Method and idealised data.

data. Hence, we abstain from even analysing the respective detection delays.

Instead, we examine the results of the other ETXBAD approach introduced in this thesis, the ”Node Picking“ approach. Figure 7.6 shows the ratios of false positives and negatives achieved with each of the discussed settings. When increasing the minReporters value, the false positive ratio for a given filter setting declines. On the other hand, the false negative ratio will also increase since the attack will no longer be reported by a sufficiently large number of nodes. Consider the results for filter set to 0.1; a false positive ratio of almost 100% in all scenarios for minReporters set to 3 indicates that this value is a very frequent observation even for non-malicious nodes. When increasing the minReporters value to 10, the false negative ratio already exhibits three outliers. When examining the movement patterns of the respective Scenarios 2, 4 and 11 (cf. Appendix A.1), it becomes apparent that the attacker, a node in the group containing nodes 21 through 30 in each of these scenarios, is neighbour to nodes in its own group only. Thus, only nine nodes would be able to report on it where some may already share perfect link with the attacker, leaving no room for an actual link quality forgery attack against them. Therefore, we expect that we cannot prevent the observed outliers from occurring without accepting a high ratio of false positives in return.

The next filter setting, 0.3 yields arguably better results concerning the false positive ratio but nevertheless the ratios obtained are not satisfying. While yielding further false negatives, in addition to the outliers discussed in the previous paragraph, false positive ratios drop to an almost acceptable with a filter setting of 0.5 for all minReporters settings except 3. Finally, with largest evaluated value for filter,0.7, the maximum false positive ratios are 0.6% when minReporters is set to 3 and 0.2% and 0.03% with values 5 and 10 respectively, indicating that nodes will rarely report forged link qualities exceeding this value for non-malicious nodes.

Unsurprisingly, the delays presented in Figure 7.7 reflect the outliers discussed above. As for

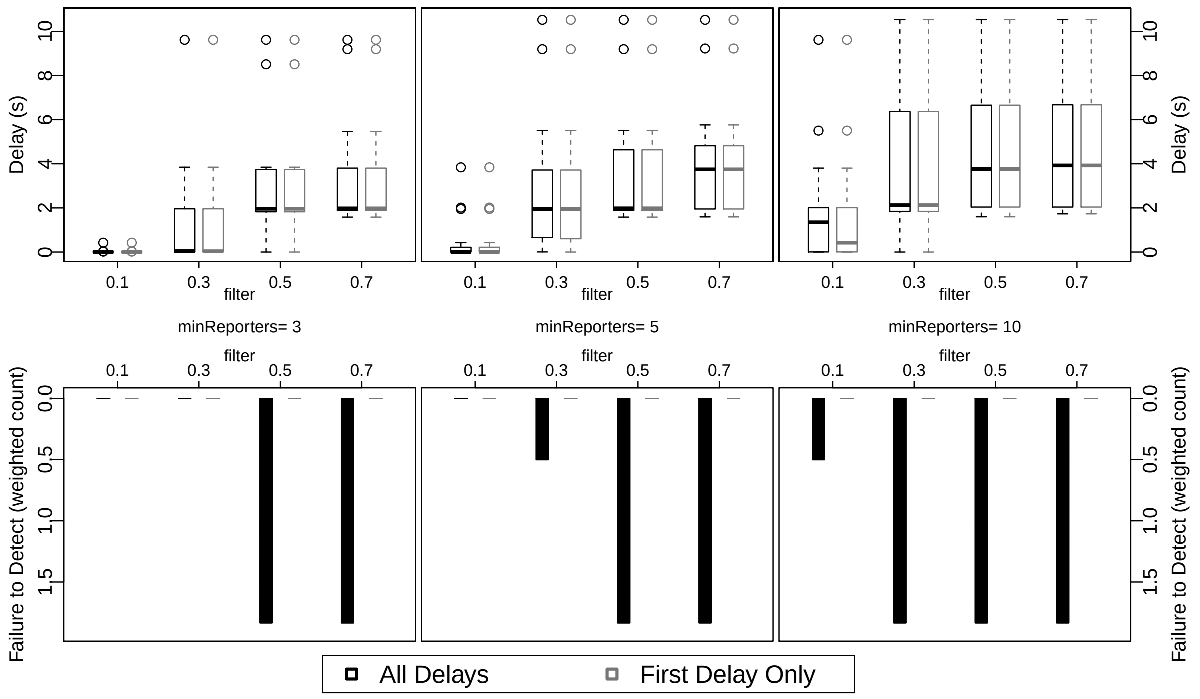


Figure 7.7.: Detection delays achieved with the ETXBAD detector when using the Node Picking method and idealised data.

the ToGBAD detector discussed in Section 7.3.2, configurations that exhibit an extreme ratio of false positives appear to result in instant or almost instant detection but are more likely to reflect the named high ratio of false positives. Below, we will only consider the configurations that yield an acceptable trade-off between false positives and negatives, i.e. where filter is either 0.5 or 0.7 and minReporters is one of 5 or 10 and the configuration where minReporters is 3 and filter is set to 0.7. Note that for each of these configurations failures to detect only occur for secondary attacks and reach exactly the same count for each of these settings. Thus, a trade-off between them is not possible and we will not discuss them in detail.

Starting with minReporters set to 5 and filter set to 0.5, we see that in 50% of the scenarios the attack is detected within roughly two seconds and after 5 seconds the attack has been detected in 75% of all scenarios. While the latter figure stays almost the same when increasing the filter setting to 0.7, only after 4 seconds would ETXBAD generate a legitimate alarm in 50% of the scenarios. The same figure is reached when minReporters is set to 10 and for filter set to either 0.5 or 0.7, however the third quartile almost reaches 7 seconds with these configurations.

Last but not least, with minReporters set to 3 and filter to 0.7, the attack is detected within just above 2 seconds in 50% of the scenarios, while after an additional 2 seconds the attack would have been detected in 75% of the scenarios. Thus, this will be our recommended setting. While the false positive ratio is slightly larger when compared against the other minReporters settings with the same filter setting, it stays well below 1%. In addition, this is counterweighted by a lower false negative ratio and shorter detection delays. Finally, we want to point out that a lower minReporters threshold implies that fewer reports suffice for generating an alarm. This may be crucial to ensure detection in the face of packet loss and a successful routing attack.

DistBAD detector evaluated parameter sets	
algorithm	advanced
method	period
autoThreshold	true
gpserror	10m
period	5 seconds
kappa	{0.2, 0.4, 0.6, 0.8, 1.0}
positionValidity	11 seconds

Other required settings (through ToGBAD main configuration)	
defaultSpeed	$2\frac{\mathrm{m}}{\mathrm{s}}$
defaultRange	350m
groupSize	10

Table 7.9.: The DistBAD detector parameter sets we will use in our evaluation. Since the DistBAD detector also depends on parameters set up through the main configuration, we provide those values in the table on the right hand side.

7.3.4. Distance Based Anomaly Detector

While the DistBAD detector provides a complex array of parameters, implementing several approaches described in 4.3, we will limit our evaluation to the most recent and promising approach which had been described in [41]. Thus, we will set the "method" parameter to "period", enabling non-link based detection and the "algorithm" setting reflecting the method for calculating distances between nodes to "advanced". To enable the feature introduced in [41] for choosing an alarm threshold automatically, we set "autoThreshold" to true. The latter approach requires that we set up "kappa", to scale the threshold determined by the algorithm. Since the algorithm determines the threshold based on the count of neighbours that appear to be legitimate and in our scenarios nodes are supposed to be neighbours to at most $\frac{2}{3}$ of the nodes in the network, i.e. at most 20 nodes (cf. Section 5.2.1), setting kappa to 0.1 or below would imply that the threshold should always remain at the static lower limit of 2. Thus we start with setting kappa to 0.2 and then explore in steps of 0.2 until reaching 1.

As for the ETXBAD approach, we will set the "period" setting to 5 seconds, i.e. the nodes' report interval and, as in [41], set the "gpserror" parameter, reflecting the assumed precision of the positioning equipment, to 10m. Since we expect to receive position updates every 5 seconds but want to be able compensate for a single lost report and minor delays, we set "positionValidity" to 11 seconds.

However, DistBAD also relies on the node's maximum speed and transmission range and also the size of a group of nodes to be set up through the global configuration. Thus, while these are not part of the DistBAD configuration we have to define these, here. For simplicity, we use the "defaultSpeed" and "defaultRange" setting to set up the respective values. We set the maximum speed to $2\frac{\mathrm{m}}{\mathrm{s}}$ ($7.2\frac{\mathrm{km}}{\mathrm{h}}$), a fast walking speed, particularly when taking into consideration that operators will usually carry a significant amount of equipment. Note that the maximum node speed used when creating the movement patterns for our evaluation was $1.5\frac{\mathrm{m}}{\mathrm{s}}$. The maximum transmission range should be 350m, since we expect that the equipment modelled by our set-up, as indicated in Section 5.2.2, will regularly cover distances of 300m but has only very low probability of covering a distance of 350m. Finally, we will set the "groupSize" parameter to 10, i.e. the number of nodes in each RPGM group as described in 5.2.1.

Evaluating DistBAD with the proposed parameter sets yielded an unpleasant surprise. Even though we were using idealised data, the left hand side of Figure 7.8 shows that even with the highest value for kappa,1.0, DistBAD would trigger alarms for up to 11% of the time frames in which no attack took place. At the same time, that setting would already result in a false negative ratio above 17% for more than half of the scenarios. When decreasing kappa to 0.8,

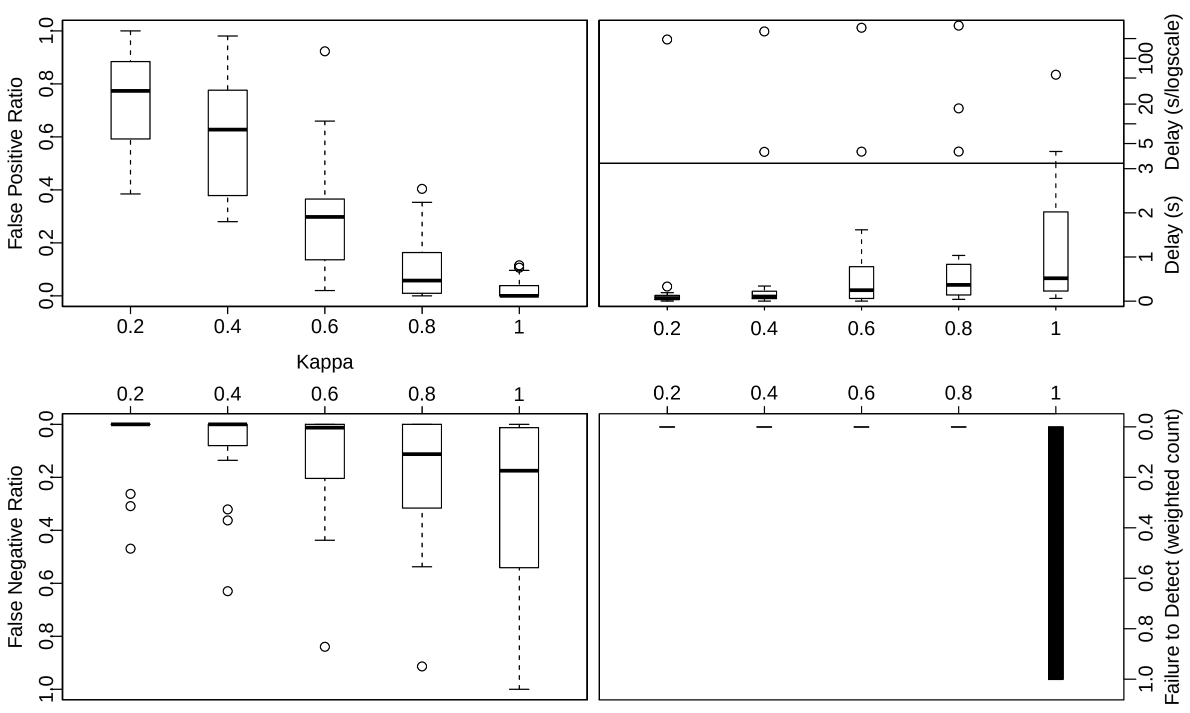


Figure 7.8.: Evaluation results for the DistBAD detector with idealised data. The left hand side of the figure shows the false positive and negative ratios, while the right hand side visualises the delays that occurred until an attack was detected. Since some extreme delays occurred, we split the respective panel on the upper right hand side into a regularly and a logarithmically scaled part.

DistBAD triggers an alarm for more than 20% of the non-attack time frames in four scenarios and only achieves a false positive ratio below 6% in only half of them. While there are almost no false negatives in one half of the scenarios at kappa $= 0.6$, this comes with a false positive ratio of just below 30% or higher for another half of the scenarios.

Lower values for kappa further decrease the ratios of false negatives, but at 0.4 the lowest false positive ratio is already at 28%, far beyond acceptable figures. We will still discuss the delays depicted on the right hand side of Figure 7.8. Since some extreme outliers occur, the upper right hand side panel, reflecting the delays, is split into a regularly scaled and a logarithmically scaled part. At the lowest value for kappa, an alarm occurs with a delay of 0.1 seconds or less for ten scenarios while at 0.4 the delay for the same scenarios reaches up to 0.2. When setting kappa to 0.6, the attack is still detected within less than a second in 75% of the scenarios. While this figure is maintained for kappa $= 0.8$, DistBAD requires up 2 seconds to reach it when kappa is at the maximum value of 1.0 and will no longer detect one attack that could previously be observed as an extreme outlier.

Figure 6.7 from Section 6.2.4 helps explain the dissatisfying performance regarding false negatives. In most scenarios, the distance between the wormhole endpoints did not exceed the maximum transmission range for more than half of the attack time frame, in Scenario 19, it was not exceeded at all. When assuming that the distance between both wormhole endpoints is exactly the maximum transmission range and they are both located in the centre of their group's radius with the other nodes in the respective groups uniformly distributed within that radius, one would expect that roughly half the tunnelled frames would be received by nodes which where within the maximum transmission range and the other half would exceed that range. While this is only a very rough approximation, it does coincide with the fact that the median of the false negative ratios is almost 0 at kappa $= 0.6$ and reaches 0 with kappa set to 0.4.

We also want to point out that the normalisation of our population hides the fact that in most scenarios the frequency of alarms was significantly higher during the attack time frames than during the non-attack time frames. While not within the scope of this thesis, it appears plausible that this could be leveraged to trade a larger detection delay against a more accurate classification.

7.4. Results

In this section, we will discuss the evaluation of the ToGBAD, ETXBAD and DistBAD detectors with the data that was available to the detector during the emulations. Other than the idealised data used in the proceeding sections, this data would be subject to delays introduced by the report interval and packet delivery but also packet loss, which may be introduced both through normal operation of a wireless network as well as by the routing attacks. We will however only present results for the configurations that promised better performance when evaluated with idealised data, as described in the previous section.

We will present results in the same order as above, starting with the ToGBAD detector in Section 7.4.1, followed by the discussion of results for the ETXBAD detector in Section 7.4.2. Finally, we will present results obtained with the DistBAD detector in Section 7.4.3.

7.4.1. Topology Graph Based Anomaly Detector

In Section 7.3.2, ToGBAD performed best with the factor setting set to 5. Thus, the results depicted in Figure 7.9 reflect the false positive and negative ratios as well as the delays and failures to detect where factor is fixed at 5. These results appear to share several characteristics with the results presented in Section 7.3.2. First of all, lower values for alpha and beta yield fewer false positives while also implying an increase in false negatives. However, in the results discussed here, the decrease in false positives is less significant, with a median at 2.8% for alpha and beta at 0.05 and at 2.5% for a value of 0.01 while it declined from 6.6% to 0.9% in the other results.

In two scenarios, Scenario 4 and 11, false negative ratios rise to unsatisfactory levels, reaching values between 44.4% and 46.9% for Scenario 4 and values between 70% and 75% for Scenario 11 as indicated by outliers in the lower left hand panel of Figure 7.9. Recalling the analysis of the sinkhole attack presented in Section 6.1.4 we note that the topology of the network allowed the attacker to be particularly successful, i.e. reduce the PDR significantly in these scenarios where the PDR had already been very low in Scenario 11 in the first place. Since this will also affect the attacked nodes' ability to deliver their report packets, the named ratios come at no surprise for these scenarios. Note that this relation also holds for other scenarios, e.g. scenario 9 yields a false negative ratio of 14.3%, 11.9% or 10.7% for alpha and beta set to 0.1, 0.3 or 0.5 respectively while scenario 1, where the attacker achieved only a modest decrease in the PDR yields a false negative ratio of 0 for each configuration.

The right hand side of Figure 7.9 shows the detection delay and failures to detect for the given configurations. As for the results generated with idealised data, the delay appears to be

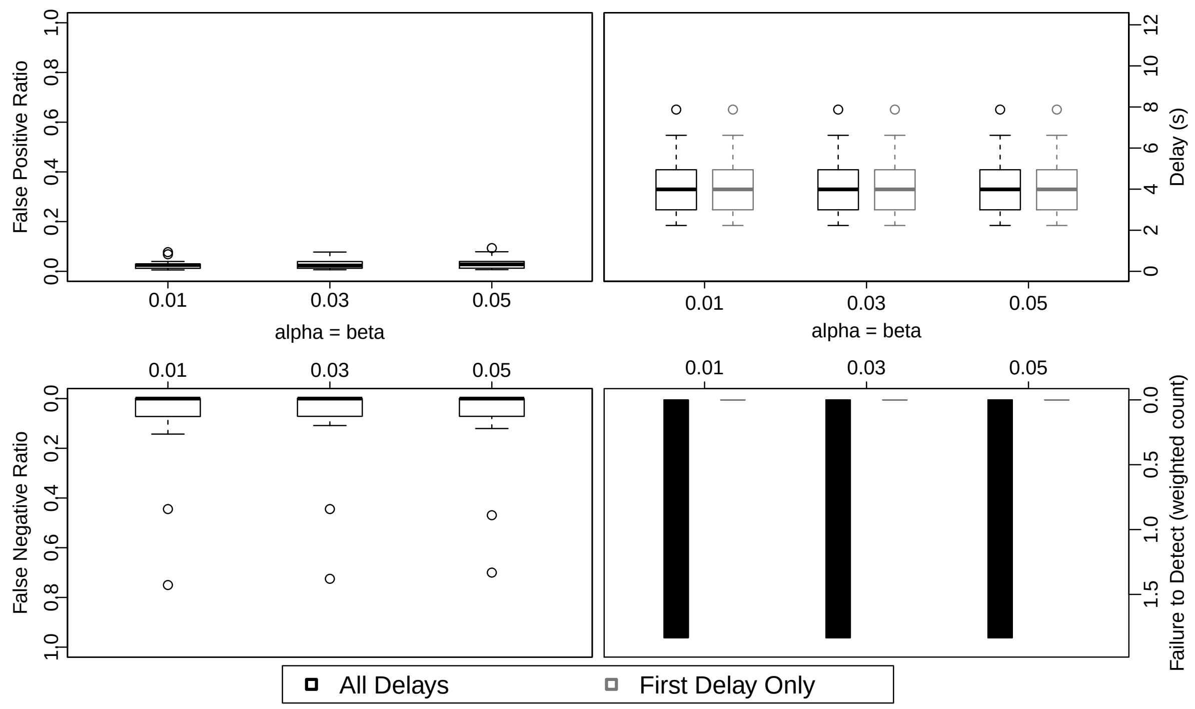


Figure 7.9.: Evaluation results for the ToGBAD detector with real data. The left hand side of the graph shows false positives and false negatives while the right hand side shows the detection delays and failures to detect an attack.

almost independent from the chosen configurations where the same is true for the count of failures to detect an attack. Since reports would be delayed here, it is not surprising that delays are larger, ranging between 2 and 8 seconds where the attack is detected within up to 5 seconds in 75% of the scenarios. While this is more than twice than respective value with idealised results, it still implies that in the large majority of the scenarios the attack would be reported within a single report interval. Note that ToGBAD appears to fail to detect any of the additional attack time frames, however the same is true for the same settings using idealised data, with the exception of a single value with weight $\frac{1}{3}$, and the arguments given in Section 6.1.4 apply respectively.

In conclusion, ToGBAD is affected by the delay and loss of reports and thus is not able to reproduce the results obtained under ideal conditions but still performs well for each of the evaluated parameter sets. While using a smaller value for alpha and beta results in a slight increase of false negative ratios and yields only a small decrease in false positive ratios, the previously suggested parameter set with a factor of 5 and alpha and beta set to 0.01 still appears appropriate since it yields the same delay but a slightly higher confidence in positive detection results when compared against the other parameter sets.

7.4.2. ETX Based Anomaly Detector

Since the ”Mean“ and ”Mean with Node Threshold“ methods performed much worse than the ”Node Picking“ method in the evaluations with idealised data presented in Section 7.3.3, we only evaluate the latter method and only with minReporters set to 3 in this section. Figure 7.10 shows the false positive and negative ratios achieved with this setting and different values

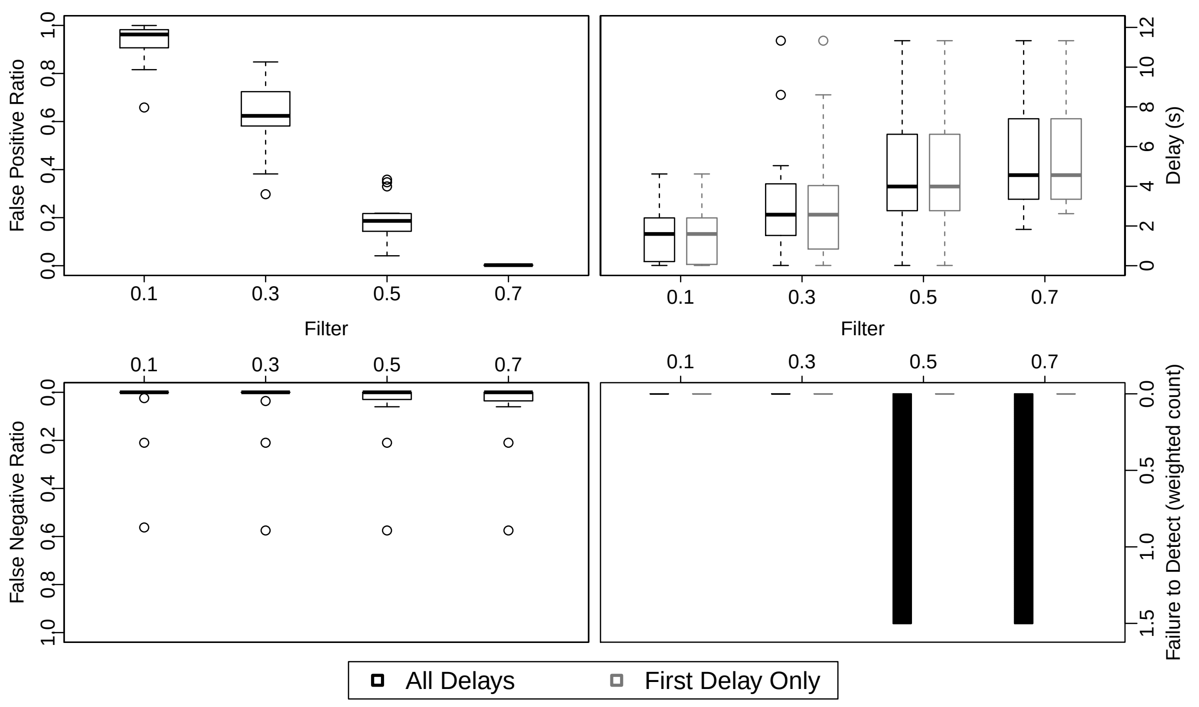


Figure 7.10.: Evaluation results for the ETXBAD detector using the Node Picking method and real data. On left hand side of the graph we present the ratios of false positives and false negatives while the right hand side depicts the detection delays and failures to detect an attack achieved with the given configurations and real data.

for the filter settings. While for any of the evaluated filter settings, the median for the false negative ratio is at 0, we observe two outliers that occur for each value of the filter setting. Our inspection of the raw data revealed that these outliers were again caused by scenarios 4 and 11 which we had already identified as the cause of similar peaks in the false negative ratio in Section 7.4.1. This indicates that the ETXBAD detector will also be affected by the loss of reports.

While another effect appears to be that fewer false positives occur for a given configuration, the respective ratios are obviously off limits for a filter setting of 0.1 or 0.3. At filter $= 0.5$ the median lies at 18.6%, indicating a sharp decline in the false positive ratio as the proceeding value was 62.4%, but is still far from satisfactory. When setting the filter to 0.7, the false positive ratio is 0.23% or less for one half of the scenarios and does not exceed 0.62 in the other. With the same setting, no false negatives occur in eight scenarios while values in the next quartile reach from 1.3% to 3.6% followed by one scenario with a false positive ratio of 6% and finally 21% and 57.5% for Scenarios 4 and 11. Since the combination of minReporters set to 3 and a filter value of 0.7 almost eliminates all false positives and still allows for satisfying false negative ratios in most scenarios, this configuration is the obvious choice for a deployment.

However, other than for the ToGBAD detector evaluated in the previous section, we will be trading accuracy for an increased detection delay. While the delay of just above 2 seconds for 75% of the scenarios when filter is set to 0.1 should not be considered competition given the implied false positive ratios, ETXBAD manages to detect some attacks instantly when the filter value is set to 0.5 but will need at least 2.6 and at most 11.3 seconds to detect the first attack time frame with our preferred value of 0.7. While it does achieve a delay of 0.87 seconds for

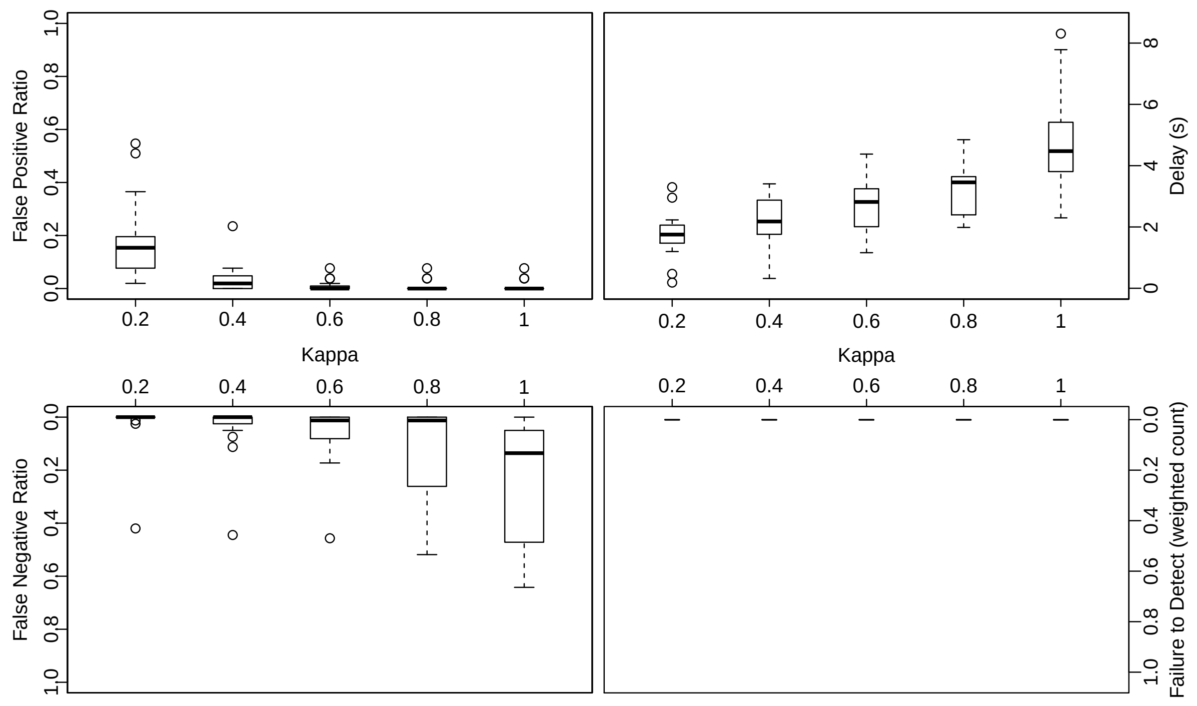


Figure 7.11.: This figure shows the results of evaluating the DistBAD detector with data obtained in emulations. The left hand side reflects the false positives and negatives obtained while the right hand side shows to the detection delay and failures to detect.

one of the latter attack time frames, it fails to detect any of the others. Given that the attack was detected in up to 4.6 seconds, i.e. less than one report interval, for one half of the scenarios we still consider the delay acceptable.

In a nutshell, we found a configuration for ETXBAD that would yield almost no false positives and low ratios of false negatives unless a large ratio of report messages was lost. While the configuration does not achieve instant detection, it never fails to detect the first attack time frame and does so with a modest delay for most scenarios.

7.4.3. Distance Based Anomaly Detector

Figure 7.11 shows the results of evaluating DistBAD with data obtained in our emulations. Obviously, these results deviate significantly from the results obtained with idealised data in Section 7.3.4. While DistBAD achieved a false positive ratio of 0 for the majority of the scenarios in those results when kappa was set to 1.0, the largest value would still reach 11.5% which is almost twice the largest value obtained with the data presented in Figure 7.11. More surprisingly, the median remains at 0 for kappa set to 0.6 or 0.8 and climbs to a modest 1.9% for a value of 0.4 where the idealised data already yielded a median at 62.7%.

Nevertheless, it is possible to explain this phenomenon when considering some details of our set up. In Section 7.3 we explained that the idealised data would contain the position of each node, generated in steps of one second. Thus, the detector would never use a position older than one second, yielding a maximum position error of 4m for node movement and 20m for inaccuracies of the positioning system, i.e. DistBAD would not generate an alarm for frames that appeared to exceed the maximum transmission range by up to 24m. On the other hand, with real data positions may be up to 11 seconds old, yielding a maximum tolerated position error of 64m. Since usually only few false positives would occur within a single time frame but DistBAD would often generate several dozens of true positives within a time frame in which the attack took place, this may be considered an indication that the frames in question simply travelled further than we would expect them to based on our configuration of the emulated wireless layer. Note that this is perfectly consistent with the respective model since it uses a probability distribution to model small scale effects that may both increase or decrease the received signal strength. If our theory was right, this would be mitigated by the larger tolerance values caused by outdated positions when using the data collected in our emulations but with our idealised data tolerance values would be too low to compensate for the extra distance travelled.

While this theory is consistent with out observations up to this point, it would imply that the detector would be more tolerant towards the difference travelled by frames which would in turn imply that the false negative ratio may remain at its previous level or even increase but cannot explain the decrease when comparing the false negative ratios shown in Figure 7.11 against those presented in Figure 7.8. Our investigation of the raw data did not reveal any clues as to what may have caused this phenomenon, however we suggest that it should be understood before fielding the application.[1]

As indicated in the previous paragraph, DistBAD achieves significantly better false negative ratios with the given data when compared against the results interpreted in Section 7.3.4. For kappa values 0.2 and 0.4 then median false negative ratio remains at 0 while rising to 1.2% for kappa = 0.6. While the median remains almost the same with a value of 1.3% for the next kappa value, several scenario exhibit a strong increase in the observed false negative ratios; with the proceeding kappa value, we observe false negative ratios above 1.3 for four scenarios only, at kappa = 0.8, seven scenarios exceed that value, five of them even exceeding a ratio of 22. Since the ratio increases further when setting kappa to 1.0 without yielding a benefit concerning the false positive ratio, this value appears even less beneficial.

The detection delays observed and presented on the upper right hand side of Figure 7.8 indicate an almost linear relation between the kappa value and the expected delay. Starting with a value for kappa of 0.2, the attack is detected within less than 2 in ten scenarios. The next value yields a median at 2.18 seconds, followed by 2.82 seconds when kappa is set to 0.6. At 0.8, the delay's median reaches 3.46 seconds and finally 4.47 with the maximum kappa value of 1.0.

While setting kappa to 0.8 or 1.0 yields almost no false positives, either value comes with a strong increase in the respective false negative ratio. On the other end of the scale, 0.2 yields almost no false negatives but also a false positive ratio above 10% for ten scenarios. Leaving us with values 0.4 and 0.6. The former however exhibits a majority of non-zero false positive ratios while the latter provides a false positive ratio of 0 for eleven out of 15 scenarios but also four scenarios with a false negative ratio of 15 or higher. Thus, we have to choose based on our own preference which is to avoid false positives and accept more false negatives as implied by choosing kappa = 0.6. Note that while this choice implies a slightly larger expected detection delay and despite the larger false negative ratio, with this choice each attack would still be

[1]We would like to emphasize on the fact that we were aware that these results look as if they were the result of a mix-up between evaluation results based on idealised and real data. However, we carefully rechecked each step of our evaluation including the respective intermediate results and could not find any evidence that such a mix-up did in fact occur.

	ToGBAD			**ETXBAD**			**DistBAD**		
Configuration	alpha = beta = 0.01 factor = 5			method = 2 filter = 0.7			kappa = 0.6		
				Results with Idealised Data					
	min	median	max	min	median	max	min	median	max
False Positives	0.41%	0.87%	2.83%	0.03%	0.26%	0.61%	2.0%	29.8%	92.3%
False Negatives	0.0%	0.0%	4.76%	0.0%	0.0%	33.8%	0.0%	0.13%	84.0%
First Detection Delay	1.56s	1.90s	2.02s	1.58s	1.98s	9.62s	0.001s	0.25s	292.6s
				Results with Real Data					
	min	median	max	min	median	max	min	median	max
False Positives	0.55%	2.17%	7.72%	0.03%	0.23%	0.62%	0.0%	0.0%	7.7%
False Negatives	0.0%	0.0%	75.0%	0.0%	0.0%	57.5%	0.0%	1.2%	45.7%
First Detection Delay	2.23s	3.99s	7.87s	2.62s	4.56s	11.33s	1.16s	2.81s	4.38s

Table 7.10.: Suggested parameters for the ToGBAD detectors. For each given metric we provide three values: The smallest value, median and largest value. Note that the table does not include the failure to detect counts since none of the suggested configurations failed to detect the first attack in any scenario and it thus would not provide any information.

detected within a single report period of 5 seconds.

7.5. Summary

In this chapter, we described our implementation of the ToGBAD concepts, including the NTS^{NG} reimplementation of the NTS programme and the extended ToGBAD application implementing the central detector for each of the ToGBAD concepts. We then defined the metrics false positives, false negatives and detection delay for our evaluation and with respect to each detector in Section 7.2.

The following Section 7.3 presented not only the configuration we considered for our evaluation but also the results obtained when executing the detectors with these results and using an idealised data set. While ToGBAD displayed promising results, depending on the configuration, ETXBAD performed very weak concerning the trade-off between false positives and false negatives when using the existing ”Mean“ method or the ”Mean with Node Threshold“ method introduced in this thesis. However, the ”Node Picking“ approach we introduced in this thesis displayed very promising results, particularly by allowing very low ratios of false positives without triggering extensive amounts of false negatives. Albeit the other detectors performed well, DistBAD not appear to allow a sane trade-off between false positives and false negatives. We assumed that this may be linked to the distance between the wormhole endpoints being less than the configured maximum transmission range throughout the larger part of most of the scenarios.

Finally, in Section 7.4 we presented evaluation results based on running the parameter sets that had performed best in the proceeding section with real data obtained in our emulations. This confirmed that the ToGBAD and ETXBAD detectors would perform very well with the selected parameter sets but suffer significant increases in false negative ratios when reports would be subject to extensive packet loss. To our surprise, DistBAD performed much better when executed on real data than it had in the evaluation with perfect data described above. While this allowed us to suggest a sane parametrisation for DistBAD, we had to emphasize that we were not able to uncover the cause for this sudden improvement and expressed that we suggest a more in-depth investigation of the phenomenon. Table 7.10 provides a quick overview to the suggested parameter sets for each detector and the evaluation results both obtained with idealised and real data for them.

Chapter 8

Conclusions and Summary

8.1. Conclusions

While the main goals for this thesis were the evaluation of the sinkhole and wormhole routing attacks in MANETs and the ToGBAD approaches for detecting them, our efforts were in every respect dominated by the challenges we faced concerning our emulation and evaluation environment. Even after investing more than half of the time scheduled for this thesis for correcting and improving the core of our emulation environment, the MES, we were only able to obtain results that would provide a modest degree of reproducibility by applying the restrictions described in Section 5.3.

To our surprise, publications concerned with other emulation software for MANETs, e.g. [5,38,54], do not describe similar issues. This may indicate that the developers did not encounter these problems, but we feel that it is more likely that they did not consider reproducibility in their evaluation process yet. Thus, we may have shed a light on an open research topic concerning MANET emulation. Since producing reliable results in our ME based environment proved to be such a challenge, we would suggest switching to alternative software if it proved to provide a higher degree of reproducibility.

Concerning the projected goals of this thesis, we were able to confirm that both the sinkhole and wormhole attack pose a significant threat to MANETs. While our metrics suggest that the wormhole attack was more successful than the sinkhole attack, we note that it also requires the dedication of more resources by the attacker, i.e. needs two nodes and an out-of-band channel. On the other hand, the wormhole attack could be perpetrated without acquiring any legitimate key material and would provide the attacker with more flexibility in choosing how to exercise the control obtained through its attack, at least when the sinkhole attacker's choice is limited to dropping packets as in OLSR based networks, as we pointed out in Section 6.1.1. Considering the goals of CNAs as laid out in [60], we conclude that an attacker capable of perpetrating a wormhole attack would choose this kind of attack rather than the less costly but also less capable sinkhole attack.

On the defensive side, the ToGBAD approaches we evaluated for this thesis provide a satisfying degree of success when detecting these attacks. ETXBAD, our implementation of the ToGBAD-LQ concept, had not performed well in a previous emulation based evaluation presented in [12], however when using the "Node Picking" method for aggregating data at the central detector that we introduced in Section 4.2.3, it would performed exceptionally well.

However, both our ToGBAD implementation and ETXBAD would suffer significant ratios of false negatives if an extensive amount of reports would be lost, indicating that additional measures to ensure report delivery may be necessary when relying on these methods for detecting attacks. We would also like to point out that with the suggested configuration ETXBAD can only detect link quality forgery if at least two nodes received forged link quality values concerning themselves, thus it appears advisable to always combine installations of ETXBAD and ToGBAD where the former solves the inability of the latter to detect forged link qualities and the latter ensures that completely forged neighbourships will not go undetected.

Finally, results concerning the DistBAD detector were mixed. This is particularly unpleasant given that it was designed to detect the wormhole attack that we identified as suited best for perpetrating CNA in MANETs. While the evaluation results presented in Section 7.4.3 are promising, they contradict much less promising results obtained with idealised data, forcing us to conclude that the detector would work better if fed with less reliable data. Since we were not able to resolve this obvious contradiction, we suggest investing further research into the question of what caused our observations. In addition to that, we presented some first thoughts on how introducing advanced refining in the DistBAD detection process might be able to mitigate the observed weaknesses in Section 7.3.4.

8.2. Summary

In this thesis we described the ToGBAD concepts for detecting forged neighbourships, link quality forgery and wormhole attacks. Since the ETXBAD implementation of the ToGBAD-LQ approach for detecting link quality forgery had not performed satisfactory in previous emulative evaluations, we introduced two additional methods for aggregating data at its central component.

We then evaluated the impact of the sinkhole and wormhole routing attacks in MANETs using emulations. Our results suggest that the success probability of the sinkhole attack shares a strong link to the topology of the attacked network, where it may achieve a very strong impact on isolated groups of nodes but only modest impact when nodes are more evenly distributed. The wormhole attack appears to be less subject to changes in topology while still some topologies, namely those where nodes have fewer neighbours, appear to be beneficial to it. However the dominant factor in achieving control over parts of a network appears to be the distance between the wormhole endpoints and the targeted nodes. We pointed out that the wormhole attack is particularly capable for perpetrating computer network attacks following the respective United States military doctrine since it provides the attacker with a range of choices on how to exercise the achieved control.

In the following emulative evaluation of our implementation of the ToGBAD approaches, we determined promising configurations where the “Node Picking” method for aggregating link quality forgery reports that we had introduced in this thesis outperformed its competition by magnitudes. However, our implementation of the ToGBAD-WH approach exhibited mixed results and surprisingly performed better when fed with lossy and delayed data. We thus suggested investing further research effort into clarification and possibly improvement of the approach.

Bibliography

[1] *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999) (2007), C1 –1184.

[2] *NATO Standardization Agreement (STANAG) 4591: The 600 Bit/s, 1200 Bit/s and 2400 Bit/s NATO Interoperable Narrow Band Voice Coder*, Tech. report, Brussels, Belgium, October 2008.

[3] *Information technology - Portable Operating System Interface (POSIX) Operating System Interface (POSIX)*, ISO/IEC/IEEE 9945 (First edition 2009-09-15) (2009).

[4] *Funkfeuer website*, 2010, www.funkfeuer.at.

[5] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, *CORE: A real-time network emulator*, Military Communications Conference, 2008. MILCOM 2008. IEEE, 2008, pp. 1 –7.

[6] A. Al-Roubaiey, T. Sheltami, A. Mahmoud, E. Shakshuki, and H. Mouftah, *AACK: Adaptive Acknowledgment Intrusion Detection for MANET with Node Detection Enhancement*, Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, 2010, pp. 634 –640.

[7] D. S. Alberts, *Getting to a 21st Century Military*, 2nd ed., Information Age Transformation, The Command and Control Research Program, 2002.

[8] N. Aschenbruck, *Realistische Modellierung von Bewegung und Datenverkehr in Katastrophenszenarien*, Ph.D. thesis, University of Bonn, January 2008.

[9] X. Ban, R. Sakar, and J. Gao, *Local Connectivity Tests to Identify Wormholes in Wireless Networks*, Proc. of the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'11), May 2011.

[10] F. Bollmann, *Kombination und Detektion von Angriffen gegen MANETs bei Verwendung einer linkqualitätsbasierten Routingmetrik*, Diploma Thesis, Rheinische Friedrich-Wilhelms-Universtiät Bonn, November 2009.

[11] T. Bosch, *Detecting Wormhole Attacks in Tactical Multi-Hop Networks*, Diploma Thesis, Rheinische Friedrich-Wilhelms-Universtiät Bonn, May 2009.

[12] J. P. Chapman, *Challenge-Response Based Detection of Link Quality Forgery in MANETs*, Lab report, Rheinische Friedrich-Wilhelms-Universtiät Bonn, November 2010.

[13] B. G. Choi, E. J. Cho, J. H. Kim, C. S. Hong, and J. H. Kim, *A sinkhole attack detection mechanism for LQI based mesh routing in WSN*, Information Networking, 2009. ICOIN 2009. International Conference on, January 2009, pp. 1 –5.

[14] S. Choi, D.-Y. Kim, D.-H. Lee, and J.-I. Jung, *WAP: Wormhole Attack Prevention Algorithm in Mobile Ad Hoc Networks*, Sensor Networks, Ubiquitous and Trustworthy Computing, 2008. SUTC '08. IEEE International Conference on, June 2008, pp. 343 –348.

[15] T. Clausen and P. Jacquet, *Optimized Link State Routing Protocol (OLSR)*, RFC 3626 (Experimental), October 2003.

[16] M. Crotti, F. Gringoli, P. Pelosato, and L. Salgarelli, *A statistical approach to IP-level classification of network traffic*, Communications, 2006. ICC '06. IEEE International Conference on, vol. 1, June 2006, pp. 170 –176.

[17] B. J. Culpepper and H. C. Tseng, *Sinkhole intrusion indicators in DSR MANETs*, Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on, October 2004, pp. 681 – 688.

[18] S. Dabideen, B. R. Smith, and J. J. Garcia-Luna-Aceves, *The Case for End-to-End Solutions to Secure Routing in MANETs*, Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th International Conference on, 2009, pp. 1 –6.

[19] E.J. Daniel and K.A. Teague, *Sensitivity of MIL-STD-3005 MELP to packet loss on IP networks*, Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on, vol. 3, aug. 2002, pp. III–77 – III–80 vol.3.

[20] D. S. J. De Couto, D. A., J. Bicket, and R. Morris, *A high-throughput path metric for multi-hop wireless routing*, Proceedings of the 9th annual international conference on Mobile computing and networking (New York, NY, USA), MobiCom '03, ACM, 2003, pp. 134–146.

[21] C. de Waal, *Bonn Mobility and Networking Suite*, 2006, web.informatik.uni-bonn.de/IV/BoMoNet/ns2.htm.

[22] H. Debar, D. Curry, and B. Feinstein, *The Intrusion Detection Message Exchange Format (IDMEF)*, RFC 4765 (Experimental), March 2007.

[23] A. Diefenbach, *Parametrisierung und simulative Evaluierung von TOGBAD*, Diploma Thesis, Rheinische Friedrich-Wilhelms-Universtiät Bonn, February 2008.

[24] J. Eriksson, S. V. Krishnamurthy, and M. Faloutsos, *TrueLink: A Practical Countermeasure to the Wormhole Attack in Wireless Networks*, Network Protocols, 2006. ICNP '06. Proceedings of the 2006 14th IEEE International Conference on, 2006, pp. 75 –84.

[25] "Förderverein Freie Netzwerke e.V.", *Freifunk website*, 2010, start.freifunk.net.

[26] D. A. Fulghum, R. Wall, and A. Butler, *Israel Shows Electronic Prowess*, Nov 2007, www.aviationweek.com/aw/generic/story.jsp?id=news/aw112607p2.xml&channel=defense.

[27] E. Gerhards-Padilla, N. Aschenbruck, and P. Martini, *TOGBAD - an approach to detect routing attacks in tactical environments*, Security and Communication Networks **4** (2011), no. 8, 793–806.

[28] ______, *Wormhole Detection using Topology Graph based Anomaly Detection*, Proc. of the 6th Workshop on Wireless and Mobile Ad-Hoc Networks, WMAN, march 2011.

[29] B. Gerner, *Graph-basierte IDS-Detektoren für taktische MANETs - Implementierung, Integration und Evaluation*, Diploma Thesis, Rheinische Friedrich-Wilhelms-Universtiät Bonn, August 2007.

[30] Gibson, J. D., *Speech coding methods, standards, and applications*, Circuits and Systems Magazine, IEEE **5** (2005), no. 4, 30 – 49.

[31] M.A. Gorlatova, P.C. Mason, M. Wang, L. Lamont, and R. Liscano, *Detecting Wormhole Attacks in Mobile Ad Hoc Networks through Protocol Breaking and Packet Timing Analysis*, Military Communications Conference, 2006. MILCOM 2006. IEEE, 2006, pp. 1 –7.

[32] J. Hommen, *Untersuchung von technischen Verfahren zur Realisierung von Frame-Kollisionen in einer MANET-Emulationsumgebung*, Master's thesis, Fachhochschule Köln, oct 2010.

[33] L. Hu and D. Evans, *Using Directional Antennas to Prevent Wormhole Attacks*, Network and Distributed System Security Symposium (NDSS 2004), 2004.

[34] Y.-C. Hu, A. Perrig, and D. B. Johnson, *Packet leashes: a defense against wormhole attacks in wireless networks*, INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, 2003.

[35] Y.-C. Hu, A. Perrig, and D. B. Johnson, *Wormhole attacks in wireless networks*, Selected Areas in Communications, IEEE Journal on **24** (2006), no. 2, 370 – 380.

[36] V. Jacobson, C. Leres, and S. McCanne, *TCPdump/libPCAP website*, 2011, www.tcpdump.org.

[37] M. Jain and H. Kandwal, *A Survey on Complex Wormhole Attack in Wireless Ad Hoc Networks*, Advances in Computing, Control, Telecommunication Technologies, 2009. ACT '09. International Conference on, December 2009, pp. 555 –558.

[38] M. A. Kaplan, T. Chen, M.A. Fecko, P. Gurung, I. Hokelek, S. Samtani, L. Wong, M. Patel, A. Staikos, and B. Greear, *Realistic wireless emulation for performance evaluation of tactical MANET protocols*, Military Communications Conference, 2009. MILCOM 2009. IEEE, 2009, pp. 1 –7.

[39] S. Keer and A. Suryavanshi, *To prevent wormhole attacks using wireless protocol in MANET*, Computer and Communication Technology (ICCCT), 2010 International Conference on, 2010, pp. 159 –163.

[40] I. Krontiris, T. Giannetsos, and T. Dimitriou, *Launching a Sinkhole Attack in Wireless Sensor Networks; The Intruder Side*, Networking and Communications, 2008. WIMOB '08. IEEE International Conference on Wireless and Mobile Computing, October 2008, pp. 526 –531.

[41] M. Lambertz, *Automatische Bestimmung eines Schwellwertes zur Erkennung von Wormhole-Angriffen*, Bachelor Thesis, Rheinische Friedrich-Wilhelms-Universtiät Bonn, Germany, 2010.

[42] M. C. Libicki, *Cyberdeterrence and Cyberwar*, RAND Corporation, Santa Monica, CA, 2009, www.rand.org/pubs/monographs/MG877.

[43] G. Lyon, *Nmap website*, 2010, www.nmap.org.

[44] V. Mahajan, M. Natu, and A. Sethi, *Analysis of wormhole intrusion attacks in manets*, Military Communications Conference, 2008. MILCOM 2008. IEEE, 2008, pp. 1 –7.

[45] S. Marti, T. J. Giuli, K. Lai, and M. Baker, *Mitigating Routing Misbehavior in Mobile Ad Hoc Networks*, International Conference on Mobile Computing and Networking, ACM, 2000, pp. 255–265.

[46] M. Medadian, M. H. Yektaie, and A. M. Rahmani, *Combat with Black hole attack in AODV routing protocol in MANET*, Internet, 2009. AH-ICI 2009. First Asian Himalayas International Conference on, November 2009, pp. 1 –5.

[47] R. Mishra, S. Sharma, and R. Agrawal, *Vulnerabilities and security for ad-hoc networks*, Networking and Information Technology (ICNIT), 2010 International Conference on, 2010, pp. 192 –196.

[48] National Imagery and Mapping Agency, *Department of Defense World Geodetic System 1984: its definition and relationships with local geodetic systems*, Tech. Report TR8350.2, National Imagery and Mapping Agency, St. Louis, MO, USA, January 2000.

[49] D.Q. Nguyen and L. Lamont, *A Simple and Efficient Detection of Wormhole Attacks*, New Technologies, Mobility and Security, 2008. NTMS '08., 2008, pp. 1 –5.

[50] U.S. Department of Defense, *United States Cyber Command*, Apr 2011, www.defense.gov/cyber.

[51] J. Orset and A. Cavalli, *A security model for olsr manet protocol*, Mobile Data Management, 2006. MDM 2006. 7th International Conference on, May 2006, p. 122.

[52] C. Pellerin, *Lynn: Cyberspace is the New Domain of Warfare*, Oct 2010, www.defense.gov/news/newsarticle.aspx?id=61310.

[53] A.A. Pirzada and C.S. McDonald, *Circumventing Sinkholes and Wormholes in Ad-hoc Wireless Networks*, International Workshop on Wireless Ad-hoc Networks, May 2005.

[54] A. Poylisher, C. Serban, J. Lee, T. Lu, R. Chadha, C.-Y.J. Chiang, K. Jakubowski, and R. Orlando, *Virtual Ad hoc Network testbeds for high fidelity testing of tactical network applications*, Military Communications Conference, 2009. MILCOM 2009. IEEE, 2009, pp. 1 –7.

[55] D. Raffo, C. Adjih, T. Clausen, and P. Mhlethaler, *An Advanced Signature System for OLSR*, Proceedings of the 2004 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04) (Washington, DC, USA), ACM Press, October 25 2004, pp. 10–16.

[56] S.S. Ramaswami and S. Upadhyaya, *Smart Handling of Colluding Black Hole Attacks in MANETs and Wireless Sensor Networks using Multipath Routing*, Information Assurance Workshop, 2006 IEEE, june 2006, pp. 253 –260.

[57] D. Sanger and E. Bumiller, *Pentagon to Consider Cyberattacks Acts of War*, May 2011, www.nytimes.com/2011/06/01/us/politics/01cyber.html.

[58] B. Schneier, *Schneier on Security: Cyberwar*, June 2007, www.schneier.com/blog/archives/2007/06/cyberwar.html.

[59] ______, *Schneier on Security: The Threat of Cyberwar Has Been Grossly Exaggerated*, July 2010, www.schneier.com/blog/archives/2010/07/the_threat_of_c.html.

[60] P. J. Schoomaker and J. B. Hudson (eds.), *Information Operations: Doctrine, Tactics, Techniques, and Procedures*, United States Army Field Manual, vol. FM 3-13, Nov 2003, www.globalsecurity.org/military/library/policy/army/fm/3-13/fm3-13.pdf.

[61] P. Sommer and I. Brown, *Reducing Systemic Cybersecurity Risk*, OECD Future Global Shocks report, 2011.

[62] R. Song and C. M. Peter, *ROLSR: A robust Optimized Link State Routing protocol for military Ad-Hoc networks*, Military Communications Conference, 2010 - MILCOM 2010, November 2010, pp. 1002 –1010.

[63] L. Tamilselvan and V. Sankaranarayanan, *Prevention of Blackhole Attack in MANET*, Wireless Broadband and Ultra Wideband Communications, 2007. AusWireless 2007. The 2nd International Conference on, 2007, p. 21.

[64] The Wireshark Foundation, *Wireshark website*, 2011, www.wireshark.org.

[65] P. Veeraraghavan and V. Limaye, *Trust in mobile Ad Hoc Networks*, Telecommunications and Malaysia International Conference on Communications, 2007. ICT-MICC 2007. IEEE International Conference on, may 2007, pp. 330 –334.

[66] M. Wang, L. Lamont, P. Mason, and M. Gorlatova, *An Effective Intrusion Detection Approach for OLSR MANET Protocol*, Secure Network Protocols, 2005. (NPSec). 1st IEEE ICNP Workshop on, 2005, pp. 55 – 60.

[67] K. S. Win and P. Gyi, *Analysis of Detecting Wormhole Attack in Wireless Networks*, Proceedings of the World Academy of Science, Engineering and Technology, vol. 48, World Academy of Science, Engineering and Technology, 2008, pp. 422–428.

[68] C. Wright, L. Ballard, S. Coulls, F. Monrose, and G. Masson, *Spot me if you can: Recovering spoken phrases in encrypted VoIP conversations*, Proceedings of IEEE Symposium on Security and Privacy (New York, NY, USA), ACM, 2008.

[69] M. Zalewski, *p0f website*, 2006, lcamtuf.coredump.cx/p0f.shtml.

[70] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas, *E-Hermes: A robust cooperative trust establishment scheme for mobile ad hoc networks*, Ad Hoc Netw. **7** (2009), 1156–1168.

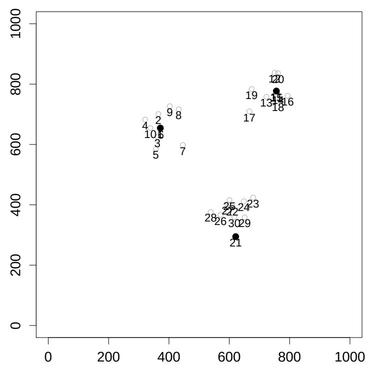
Appendix A

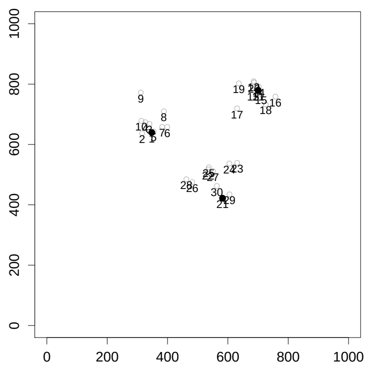
Scenarios

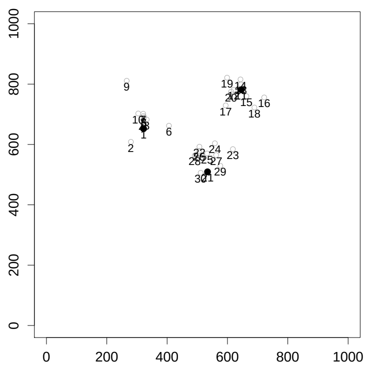
A.1. Movement Patterns

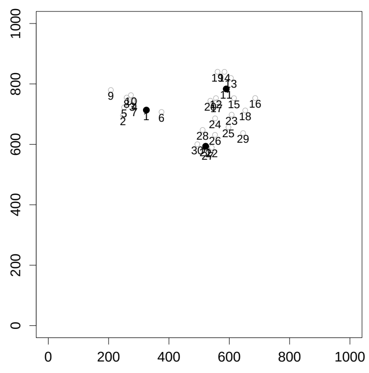
The graphs in this section display the node positions in the virtual plane. Since node positions are not static, we show their positions in steps of 100 seconds beginning with second 0 of the emulation time. Note that nodes 1 through 10 are members of the first RPGM group while nodes 11 through 20 and 21 through 30 belong to the second and third RPGM group respectively. Nodes participating in the command and control communication group are printed as solid black dots while all other nodes are printed as grey circles. Both x and y axis reflect the respective position in meters.

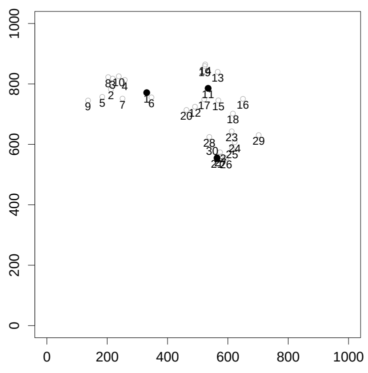
Scenario 0

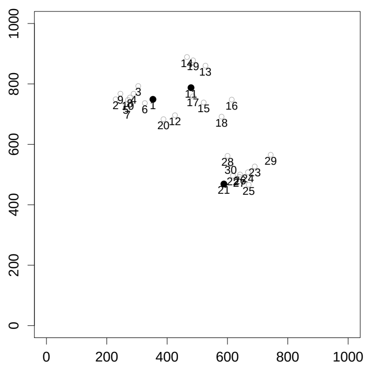

Second 0

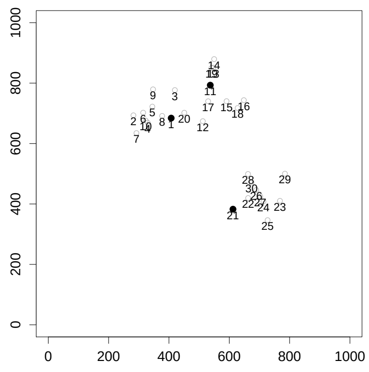

Second 100

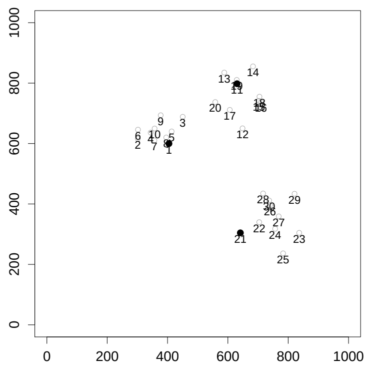

Second 200

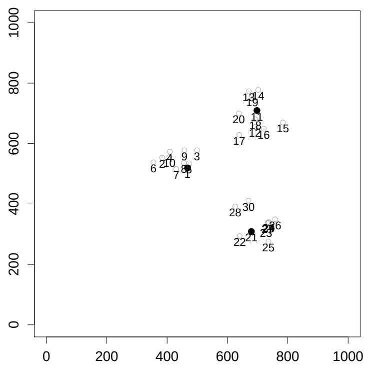

Second 300

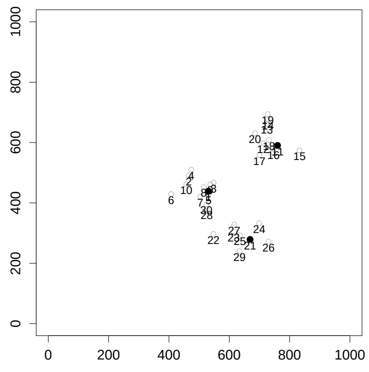

Second 400

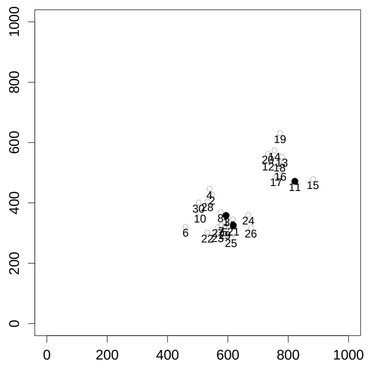

Second 500


Second 600

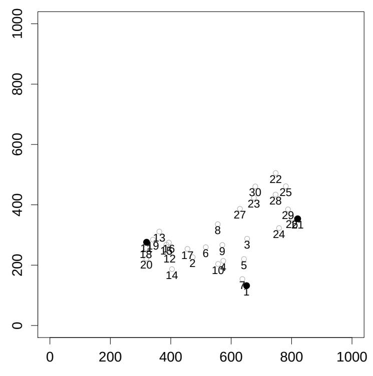

Second 700

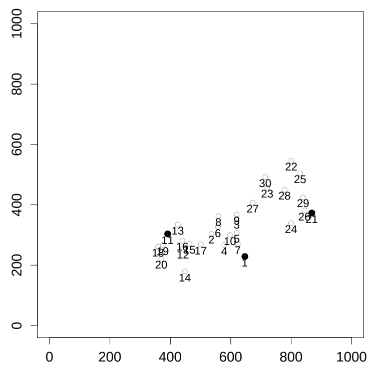

Second 800

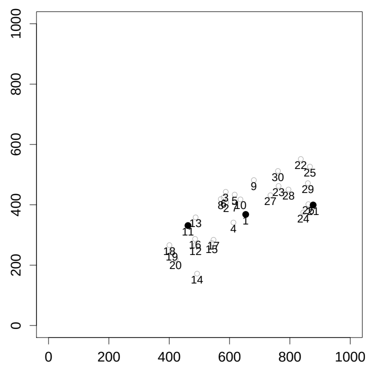

Second 900

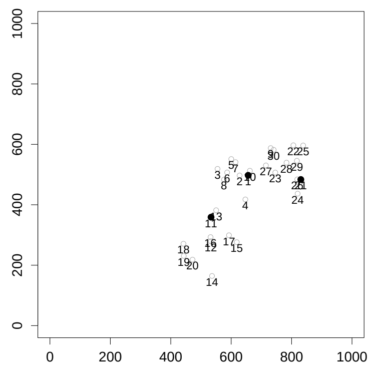

Second 1000

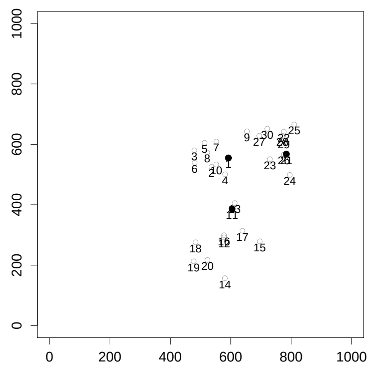
Scenario 1

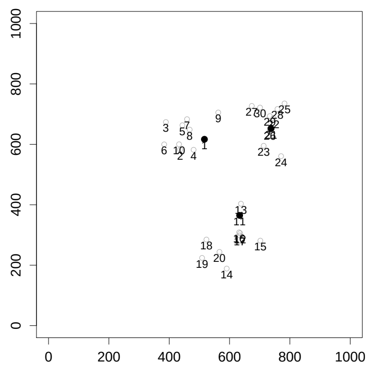

Second 0

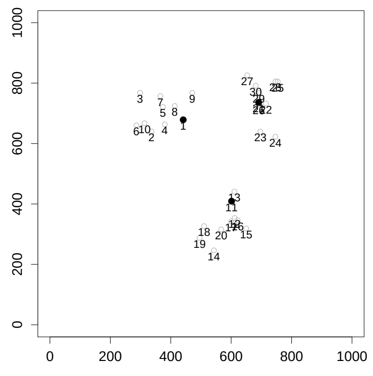

Second 100

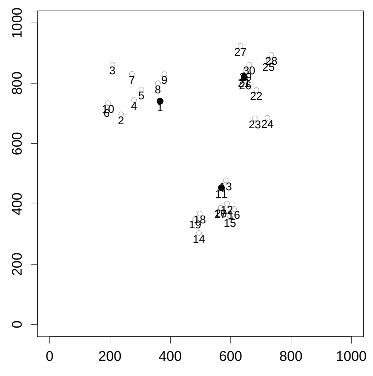

Second 200

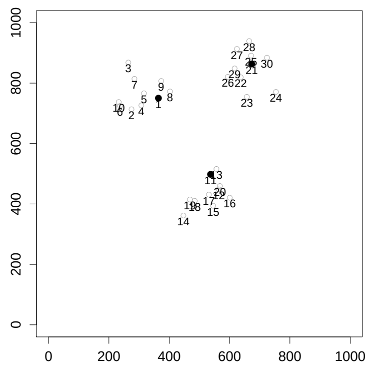

Second 300

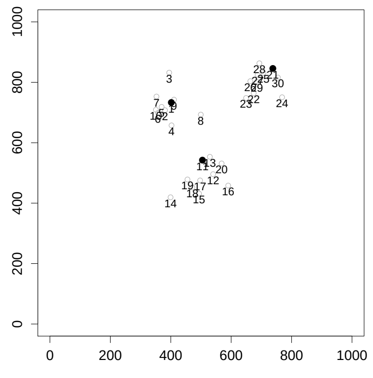

Second 400

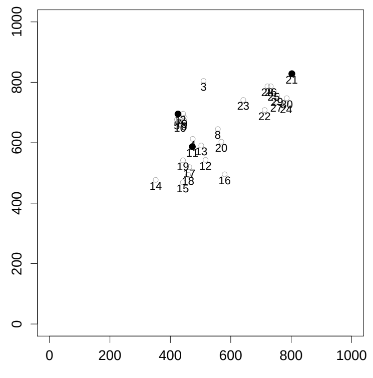

Second 500


Second 600

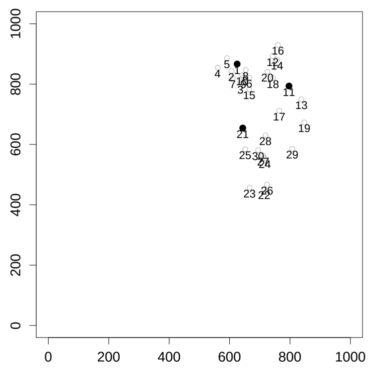

Second 700

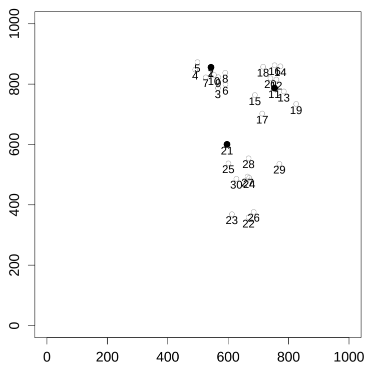

Second 800

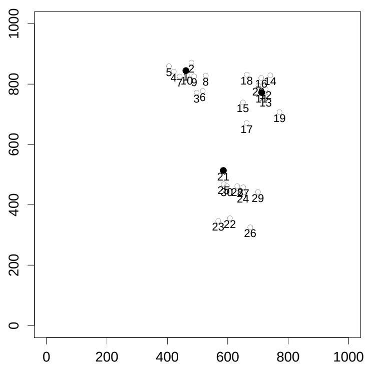

Second 900

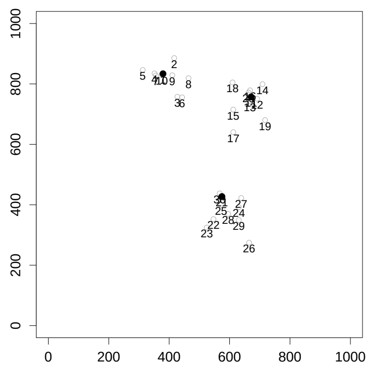

Second 1000

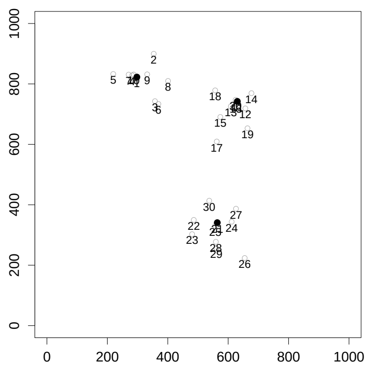
Scenario 2

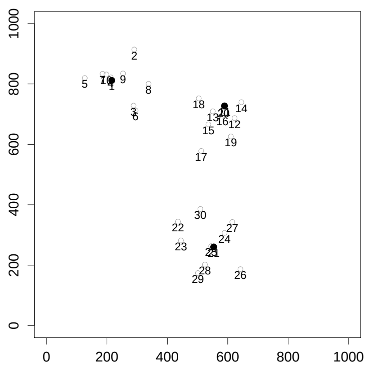

Second 0

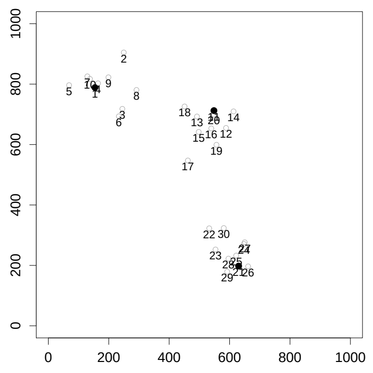

Second 100

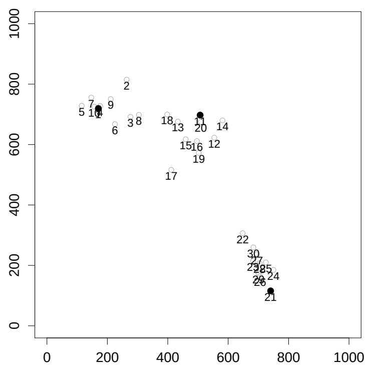

Second 200

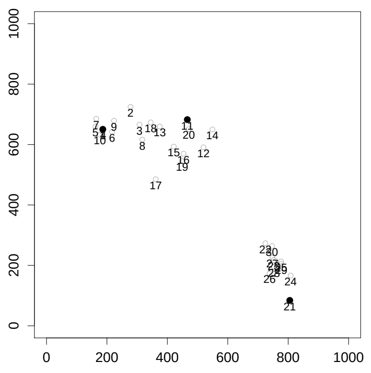

Second 300

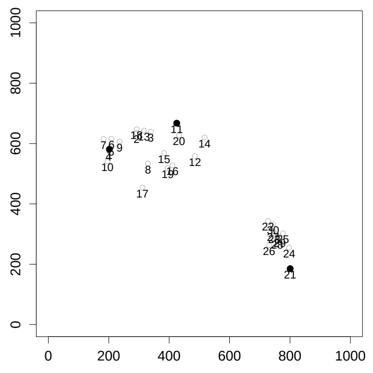

Second 400

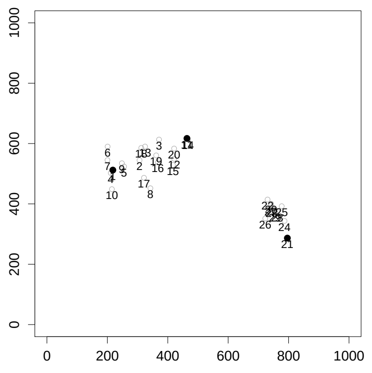

Second 500


Second 600

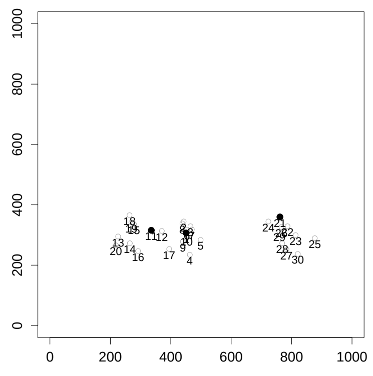

Second 700

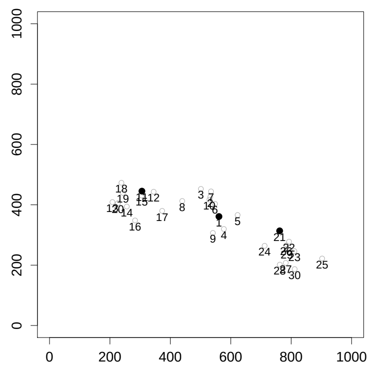

Second 800

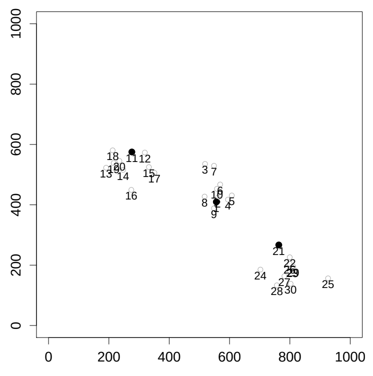

Second 900

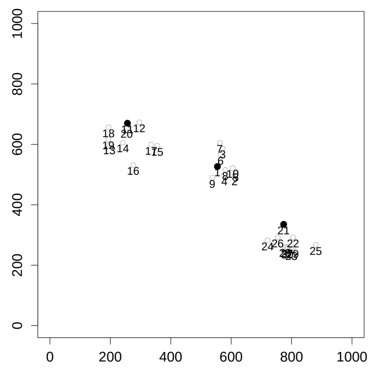

Second 1000

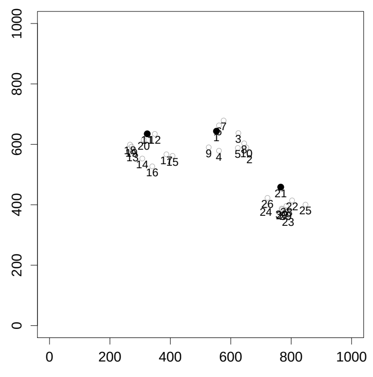
Scenario 3

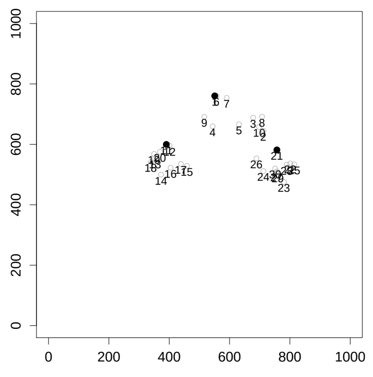

Second 0

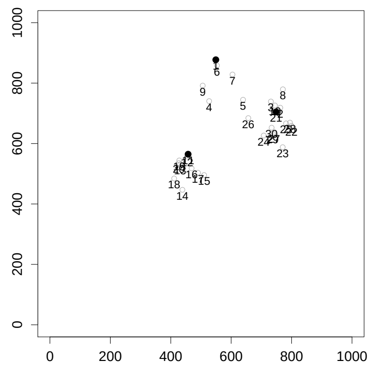

Second 100

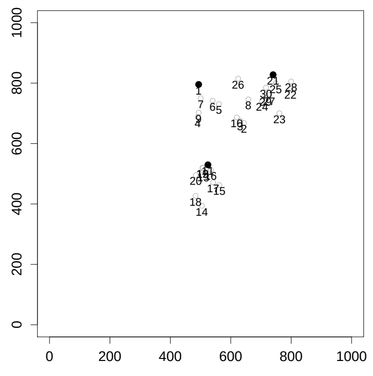

Second 200

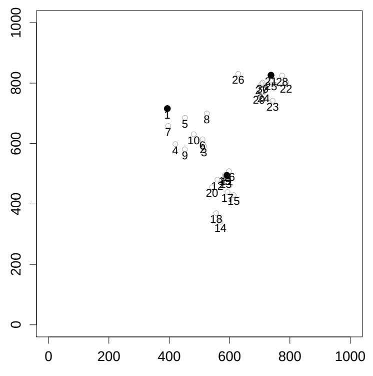

Second 300

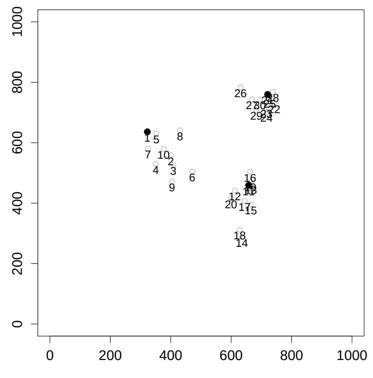

Second 400

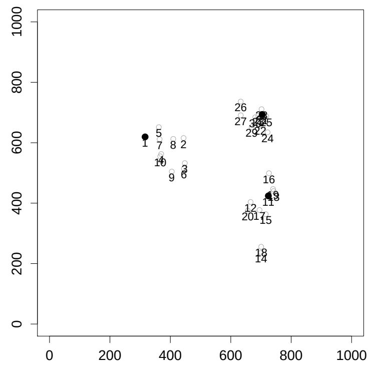

Second 500


Second 600

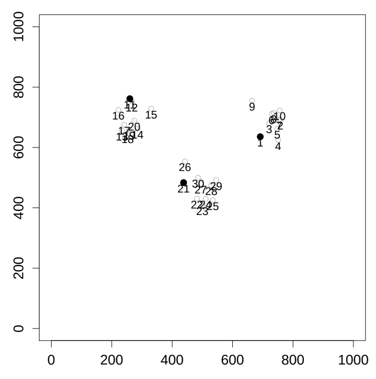

Second 700

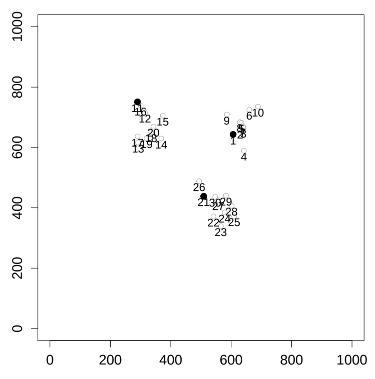

Second 800

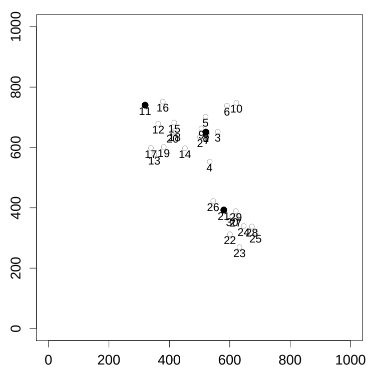

Second 900

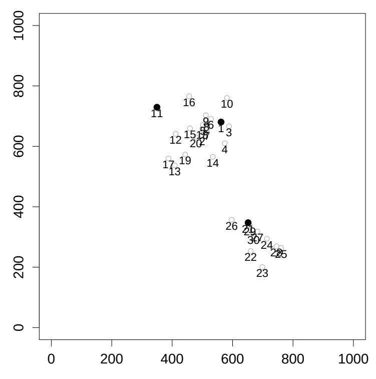

Second 1000

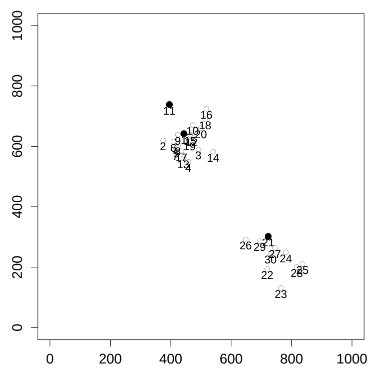
Scenario 4

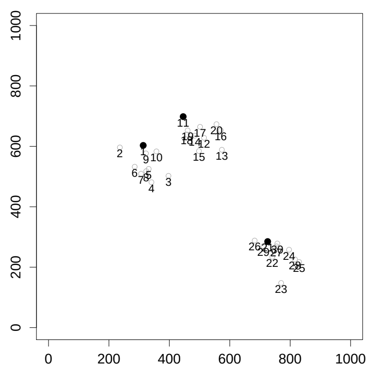

Second 0

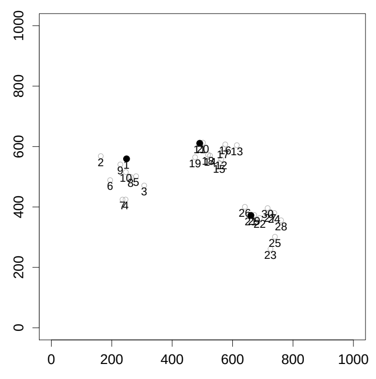

Second 100

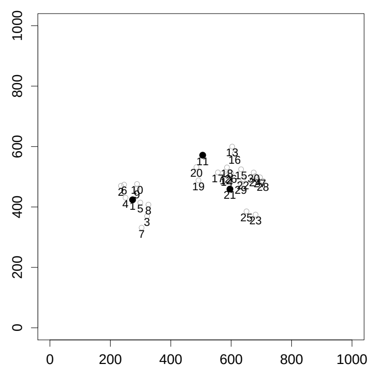

Second 200

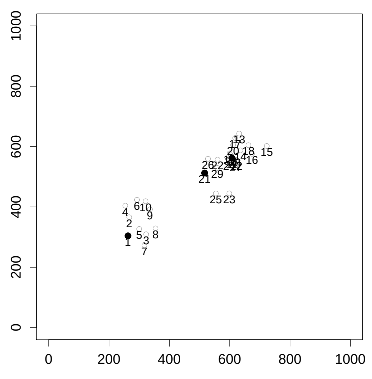

Second 300

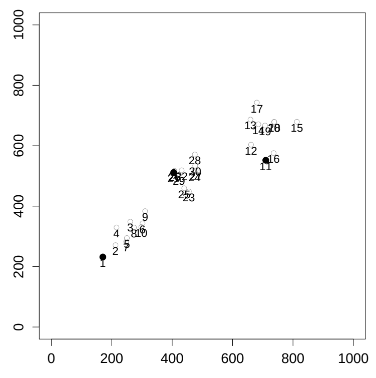

Second 400

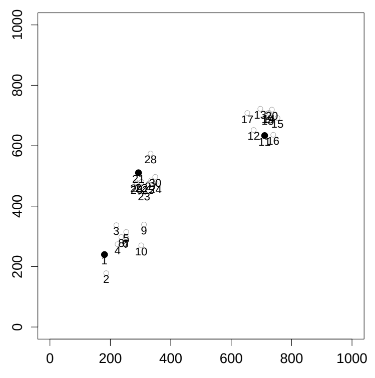

Second 500


Second 600

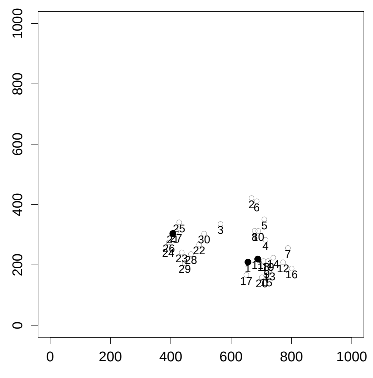

Second 700

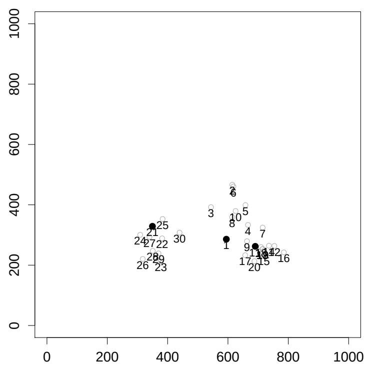

Second 800

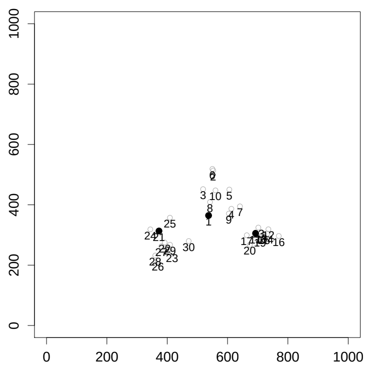

Second 900

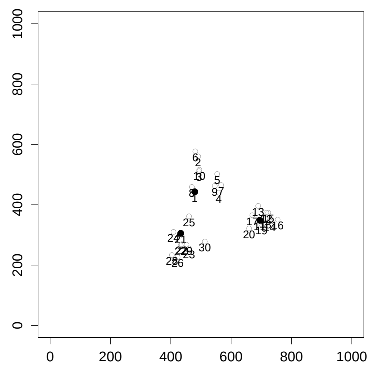

Second 1000

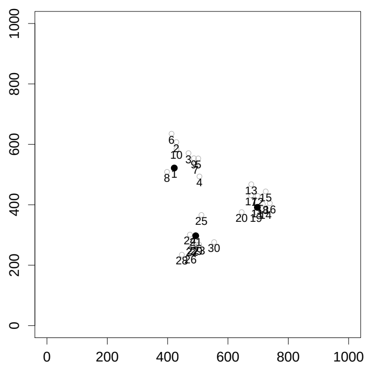
Scenario 5

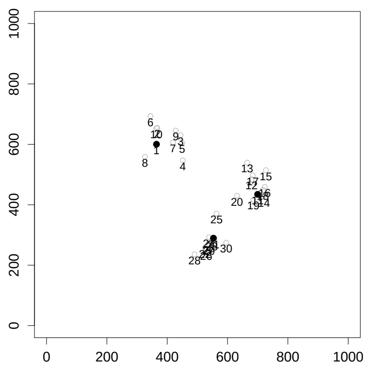

Second 0

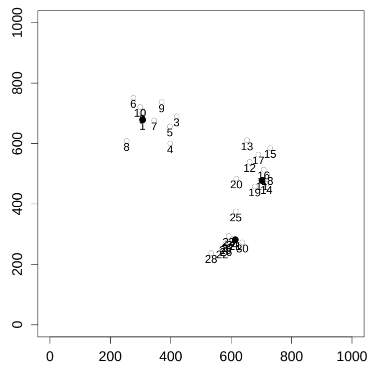

Second 100

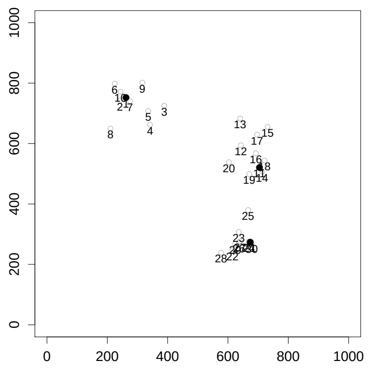

Second 200

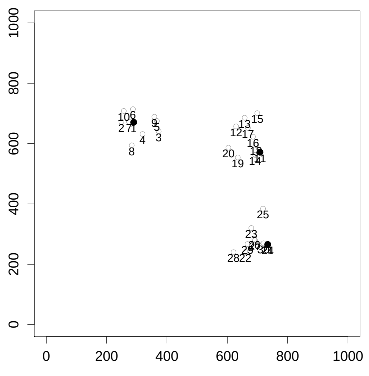

Second 300

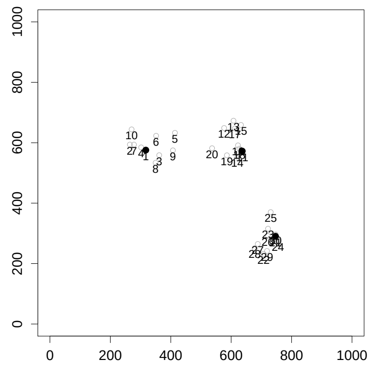

Second 400

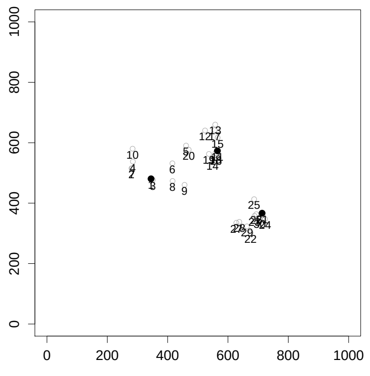

Second 500


Second 600

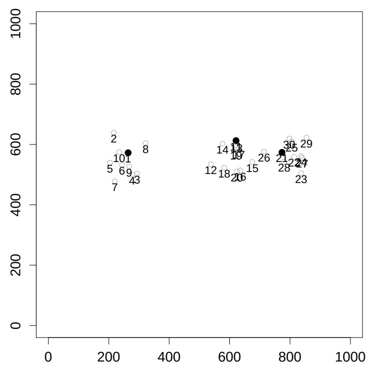

Second 700

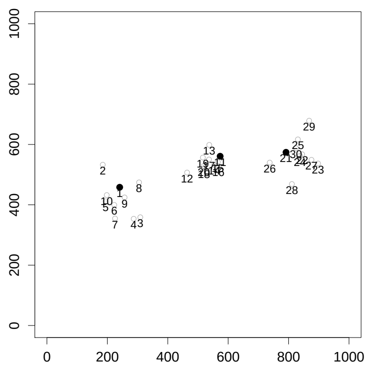

Second 800

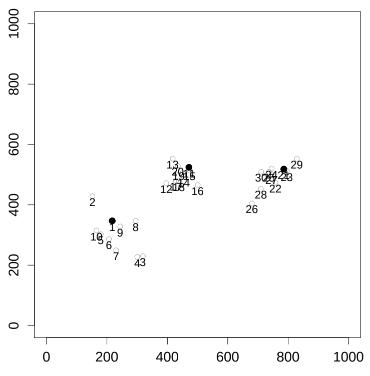

Second 900

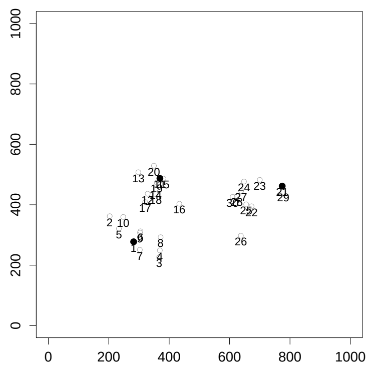

Second 1000

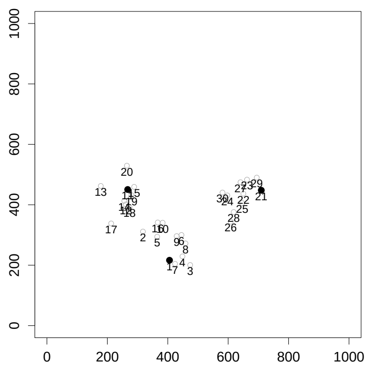
Scenario 6

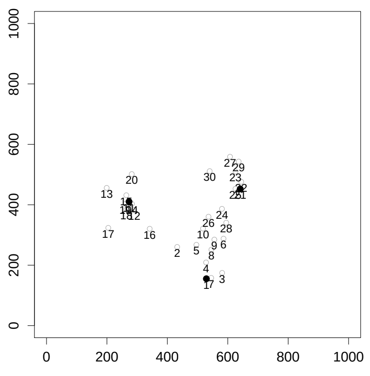

Second 0

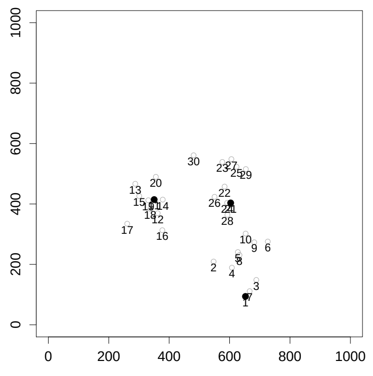

Second 100

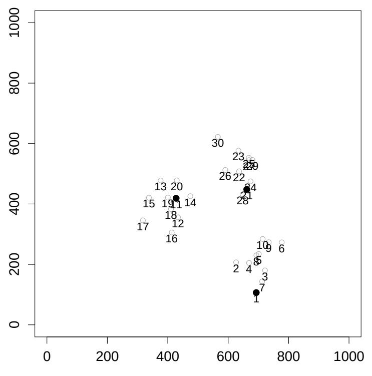

Second 200

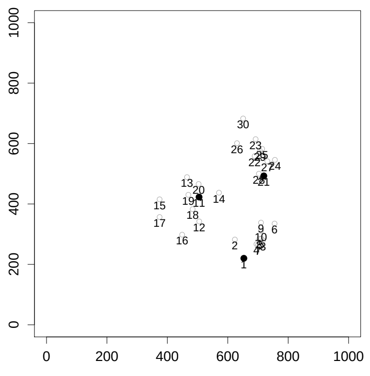

Second 300

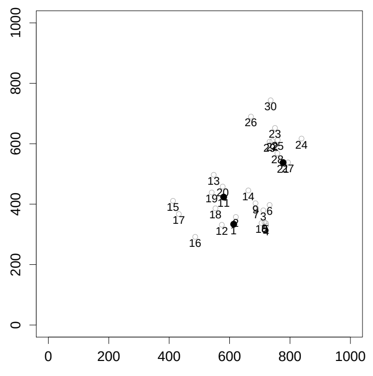

Second 400

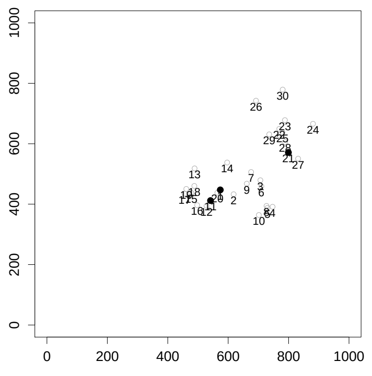

Second 500


Second 600

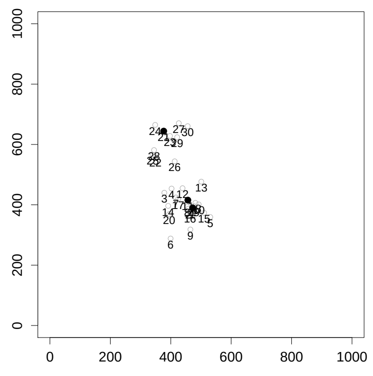

Second 700

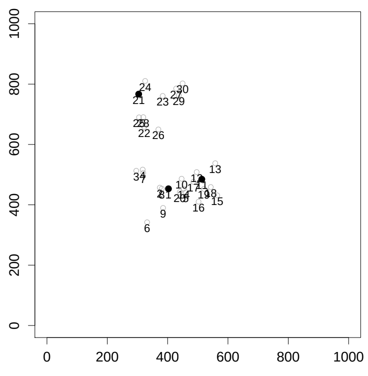

Second 800

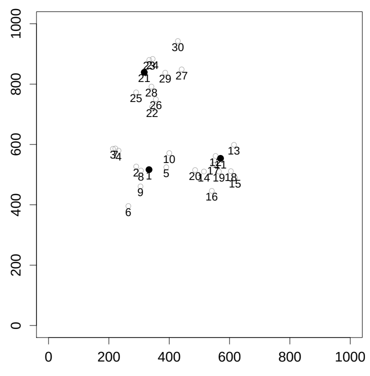

Second 900

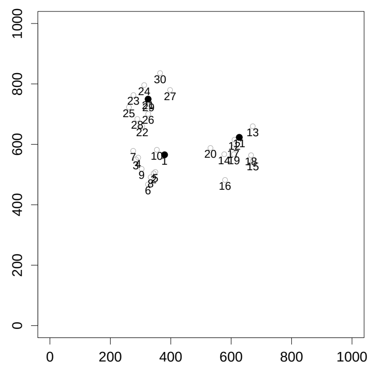

Second 1000

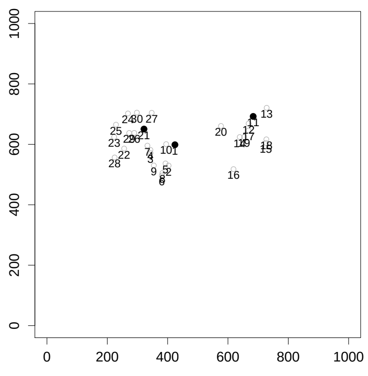
Scenario 7

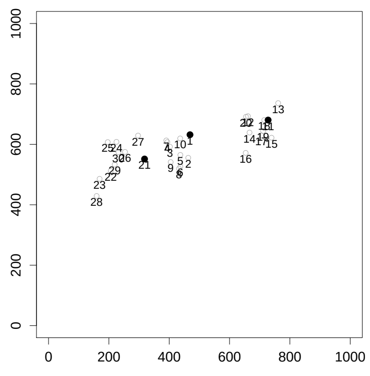

Second 0

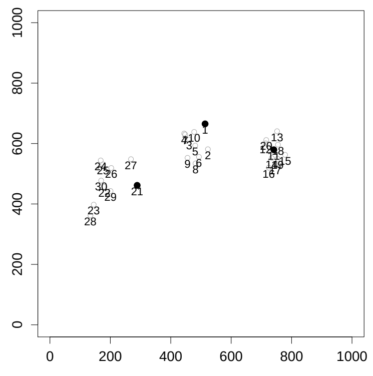

Second 100

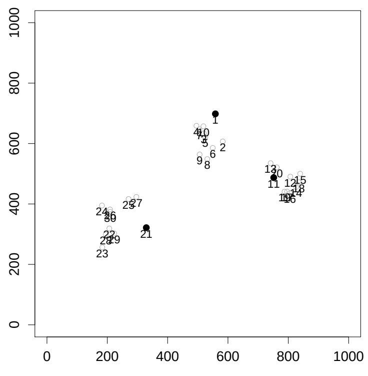

Second 200

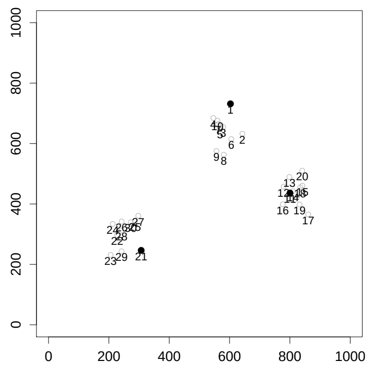

Second 300

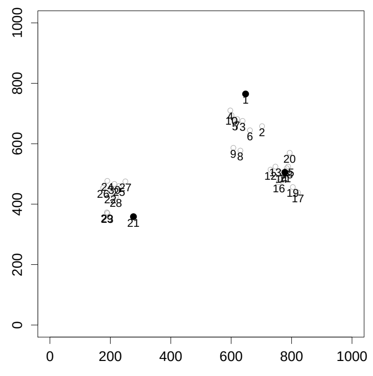

Second 400

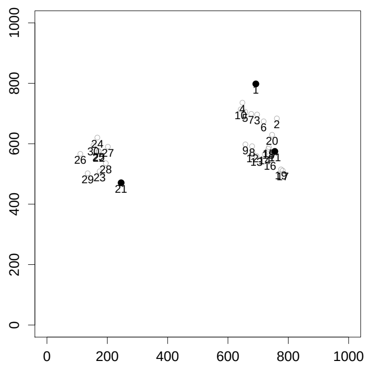

Second 500


Second 600

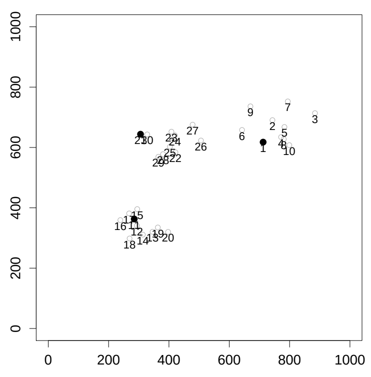

Second 700

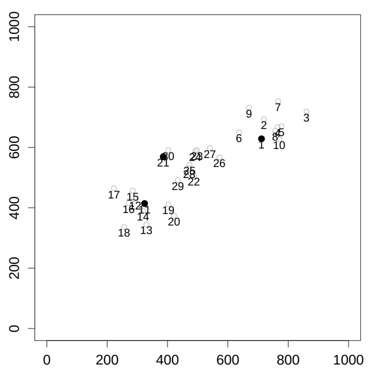

Second 800

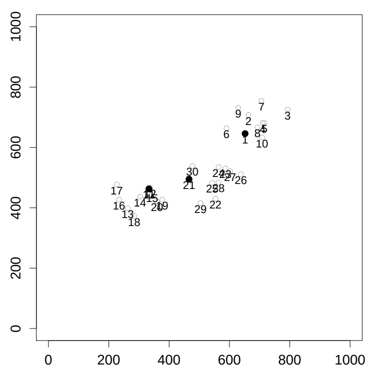

Second 900

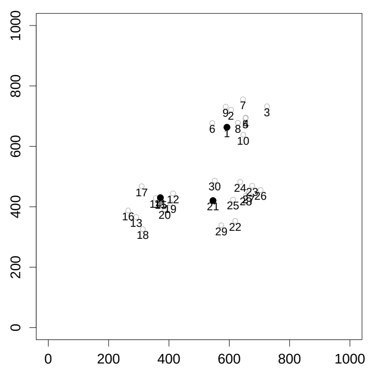

Second 1000

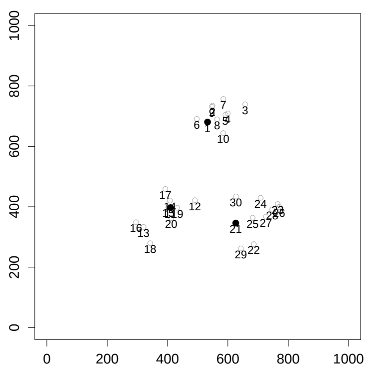
Scenario 8

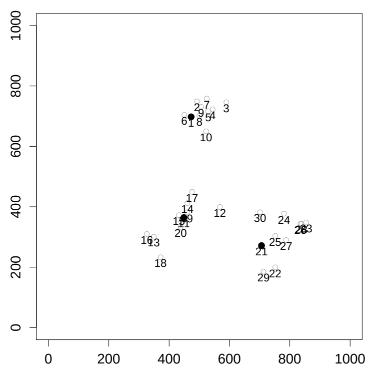

Second 0

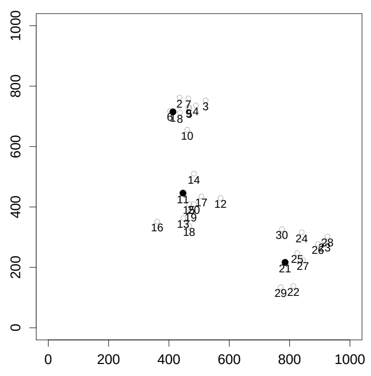

Second 100

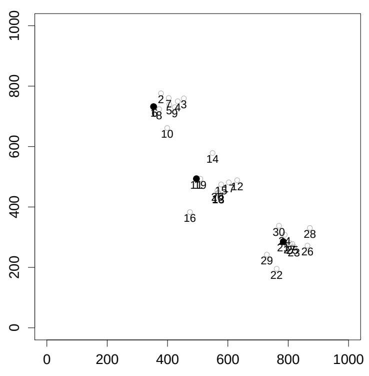

Second 200

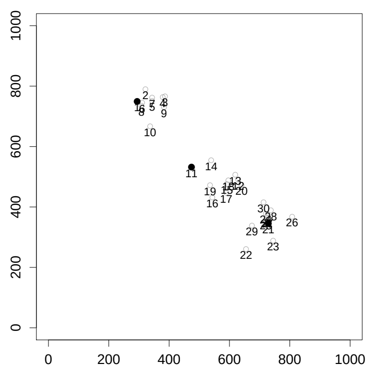

Second 300

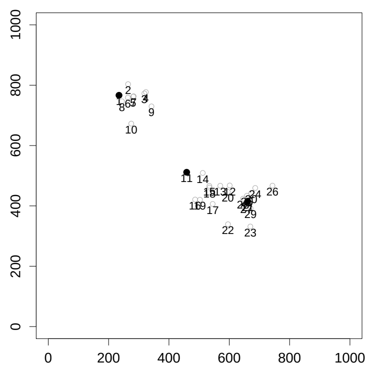

Second 400

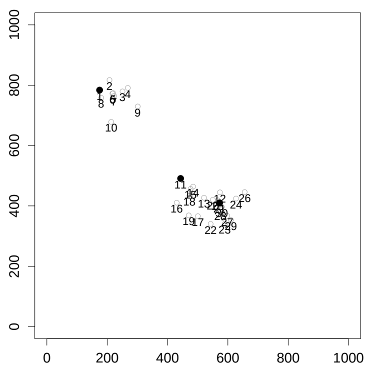

Second 500


Second 600

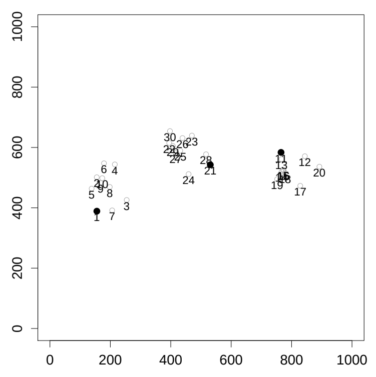

Second 700

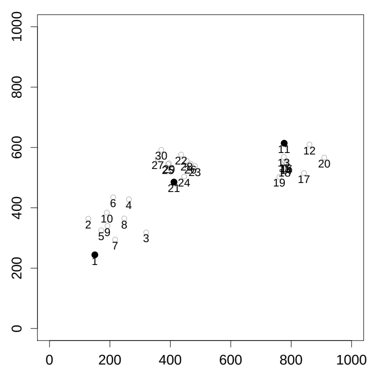

Second 800

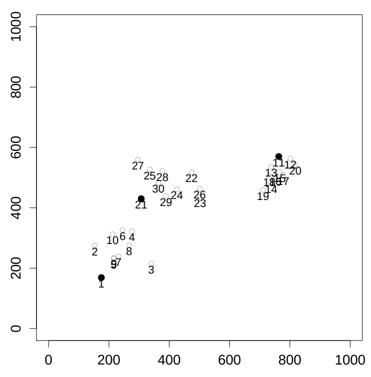

Second 900

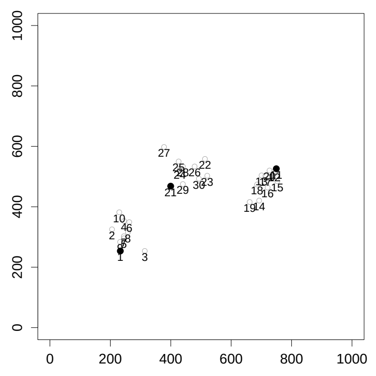

Second 1000

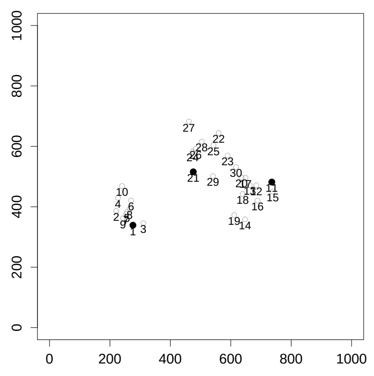
Scenario 9

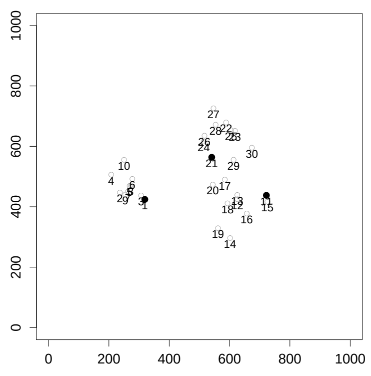

Second 0

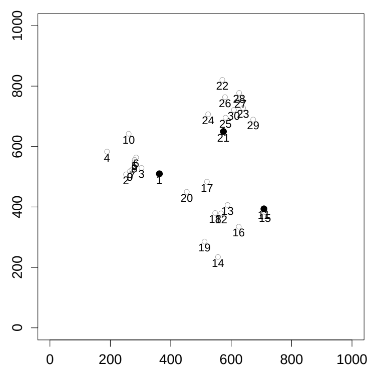

Second 100

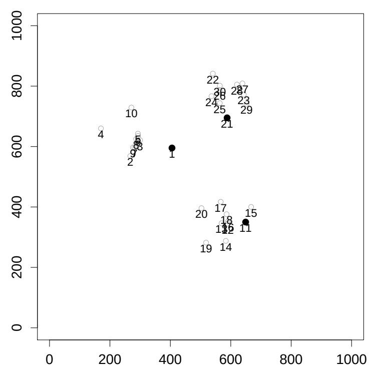

Second 200

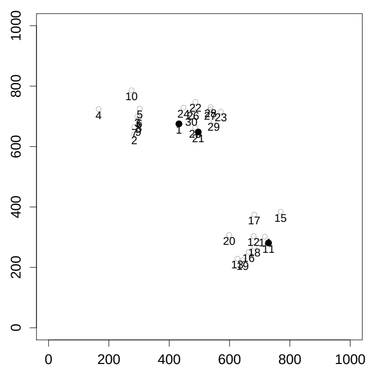

Second 300

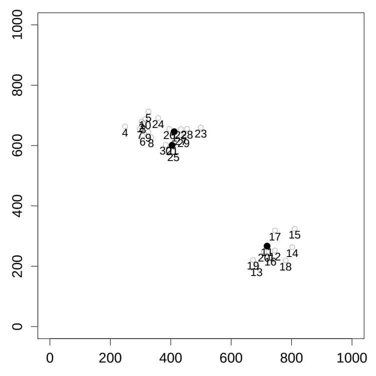

Second 400

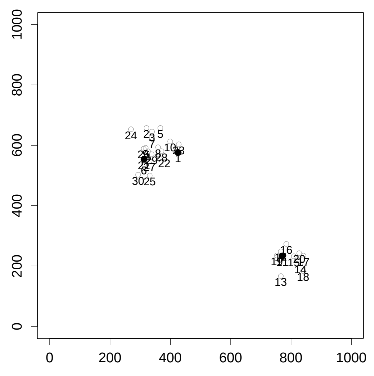

Second 500


Second 600

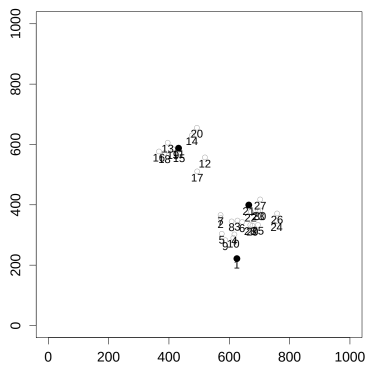

Second 700

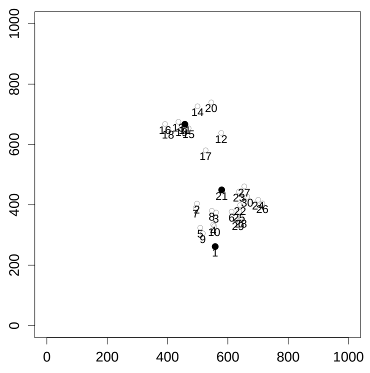

Second 800

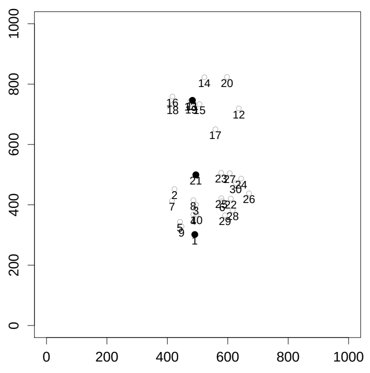

Second 900

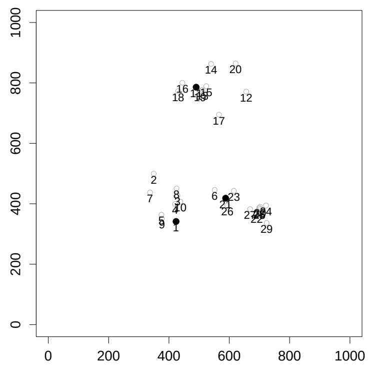

Second 1000

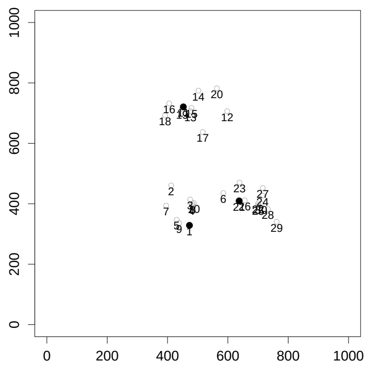
Scenario 10

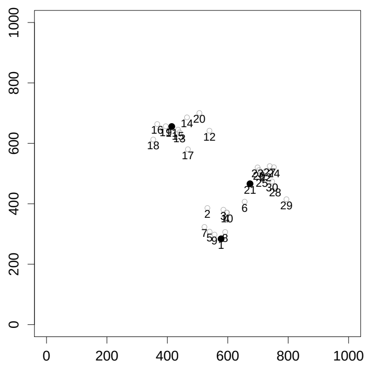

Second 0

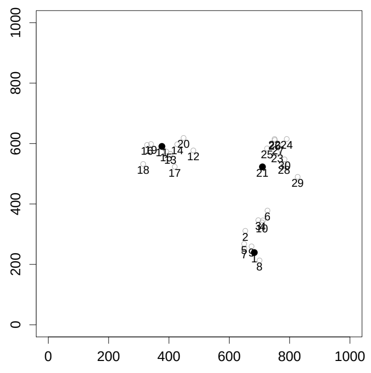

Second 100

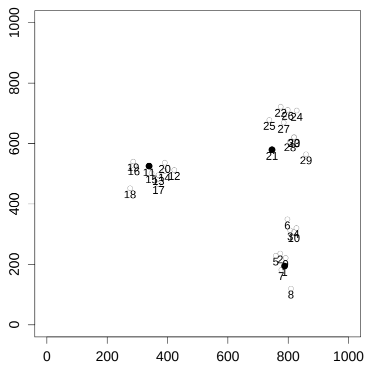

Second 200

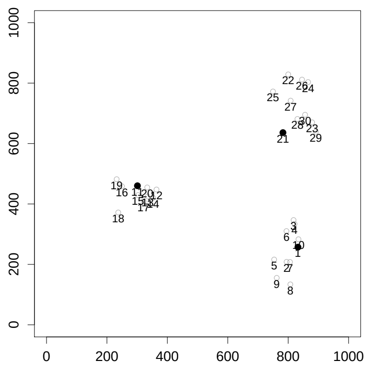

Second 300

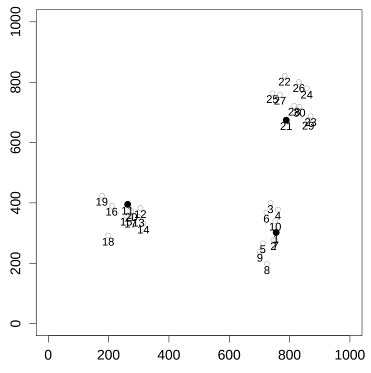

Second 400

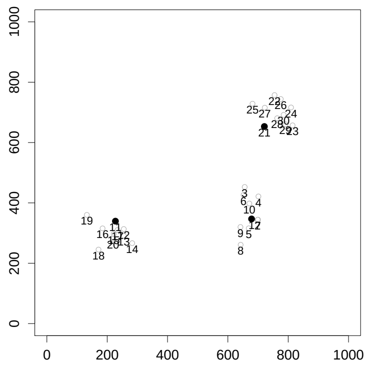

Second 500


Second 600

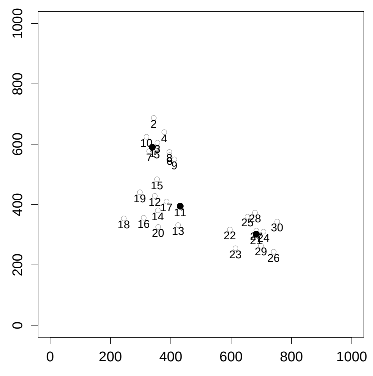

Second 700

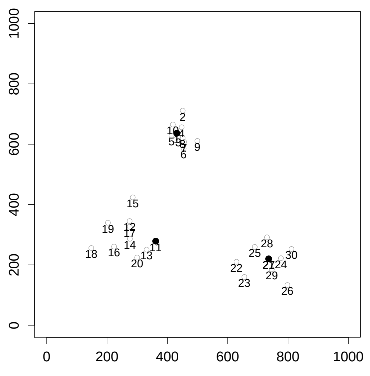

Second 800

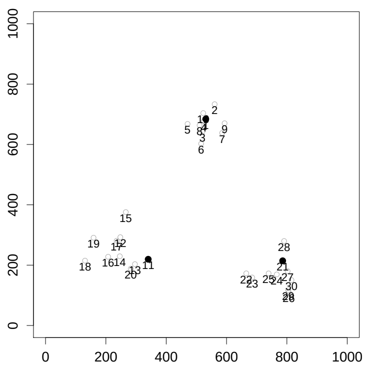

Second 900

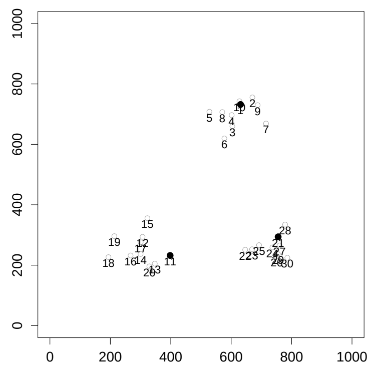

Second 1000

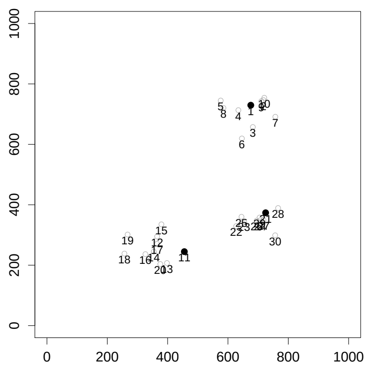
Scenario 11

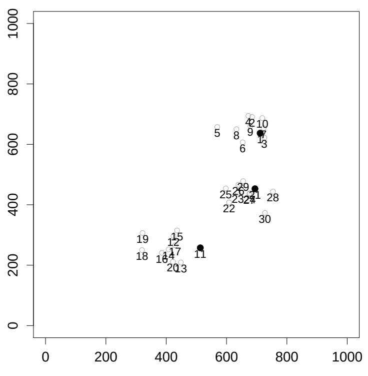

Second 0

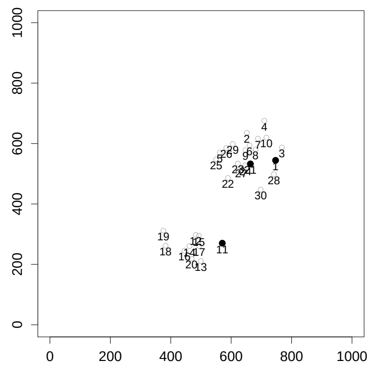

Second 100

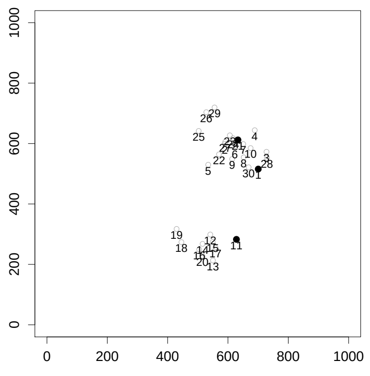

Second 200

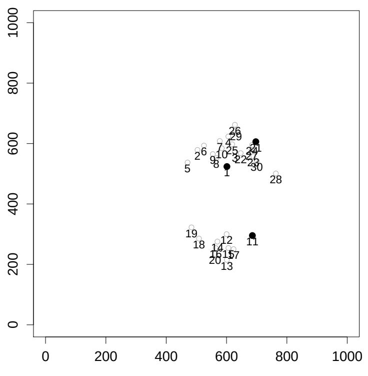

Second 300

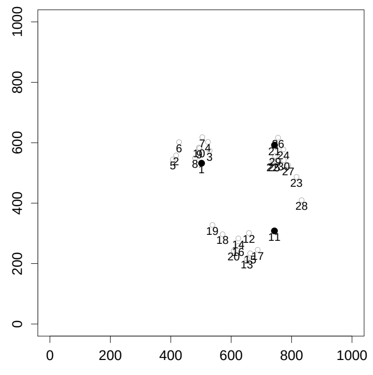

Second 400

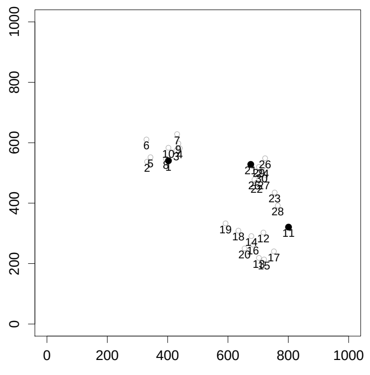

Second 500


Second 600

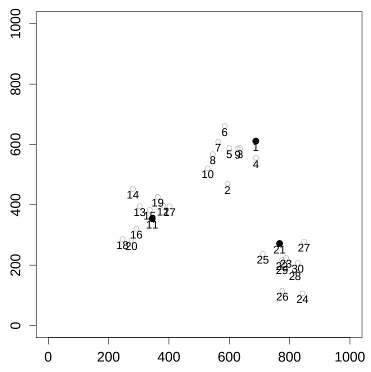

Second 700

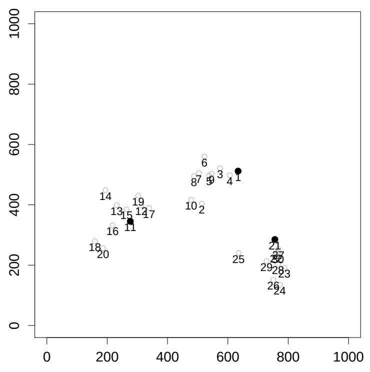

Second 800

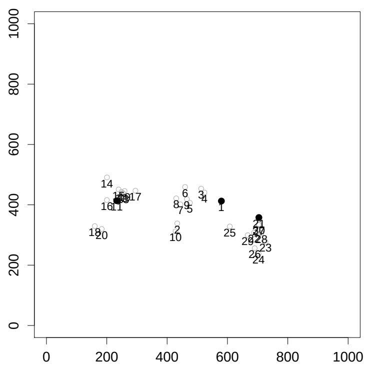

Second 900

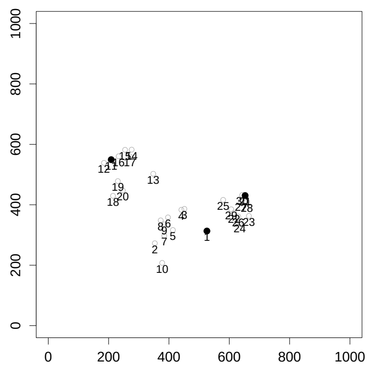

Second 1000

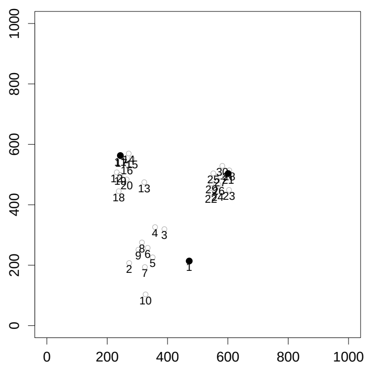
Scenario 12

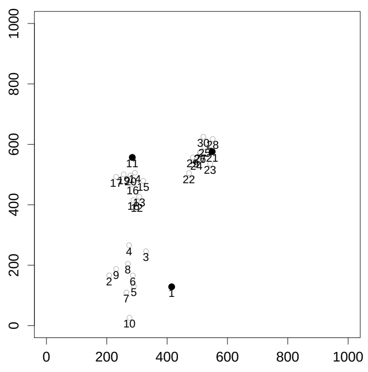

Second 0

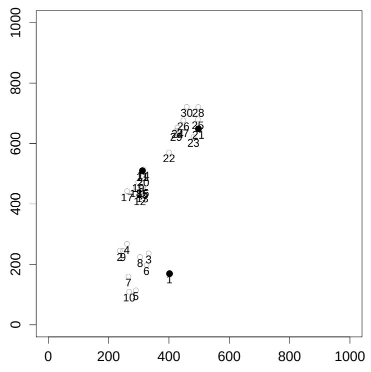

Second 100

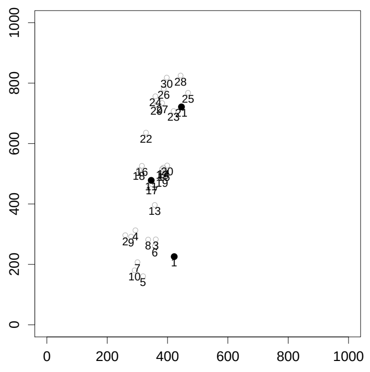

Second 200

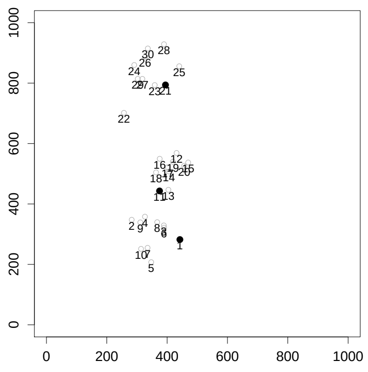

Second 300

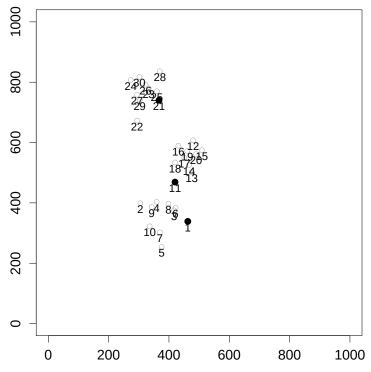

Second 400

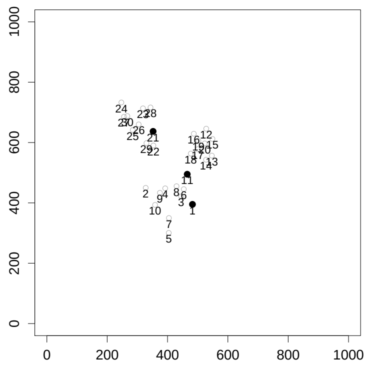

Second 500


Second 600

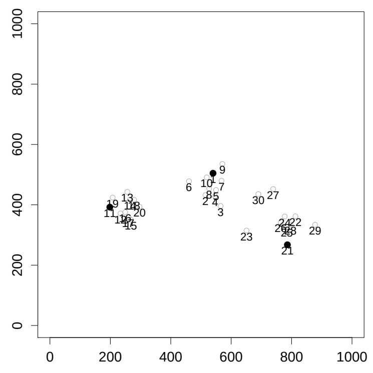

Second 700

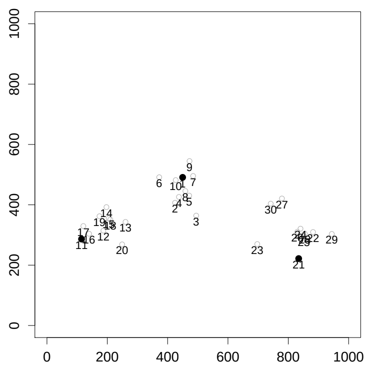

Second 800

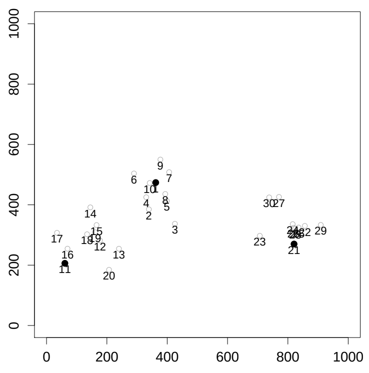

Second 900

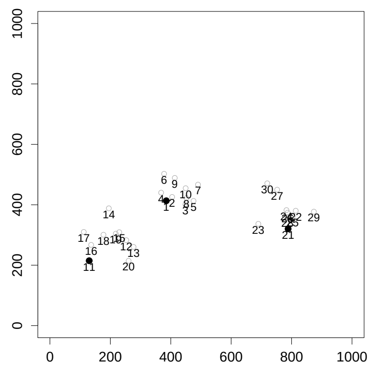

Second 1000

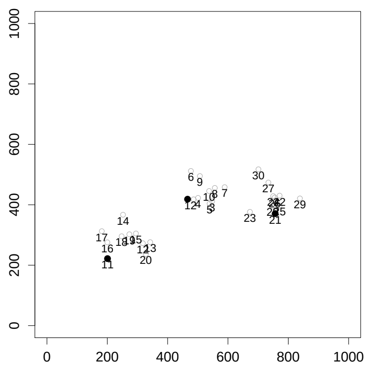
Scenario 13

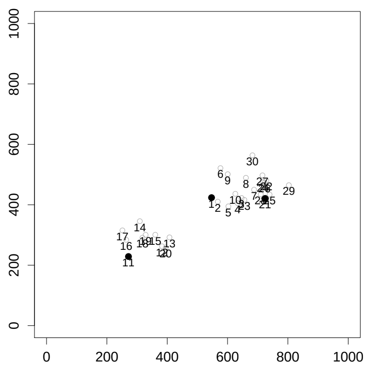

Second 0

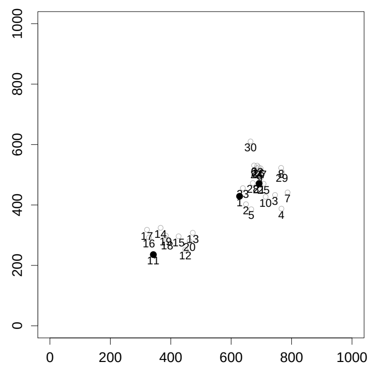

Second 100

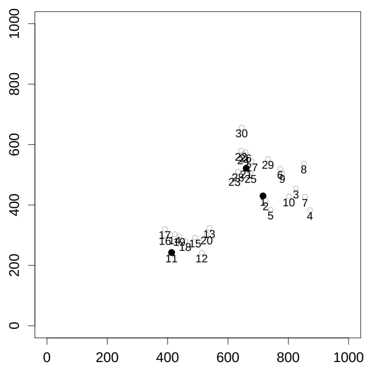

Second 200

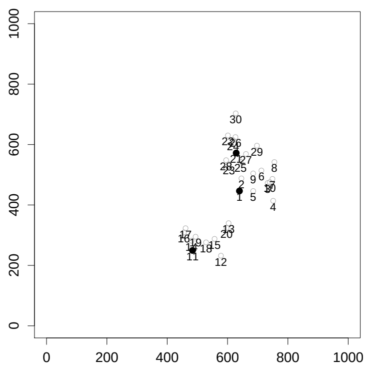

Second 300

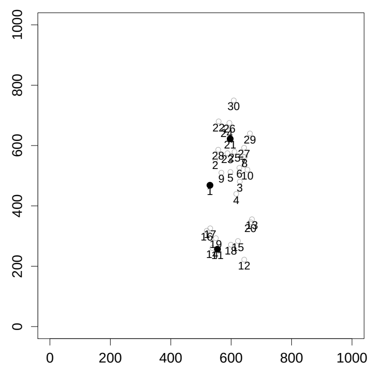

Second 400

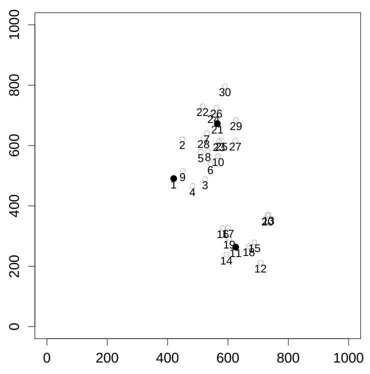

Second 500


Second 600

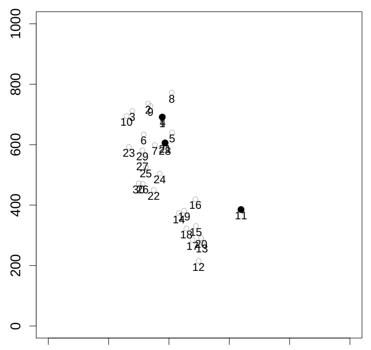

Second 700

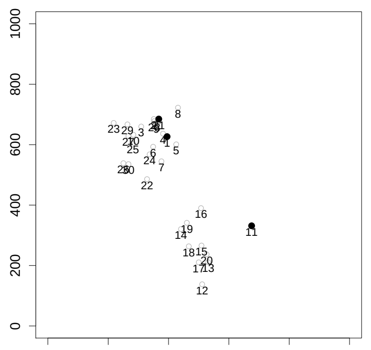

Second 800

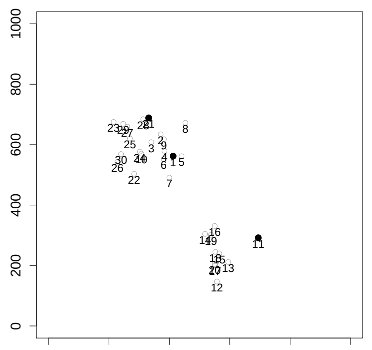

Second 900

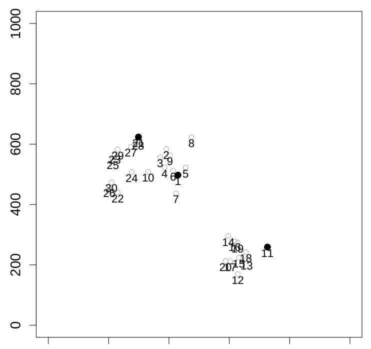

Second 1000

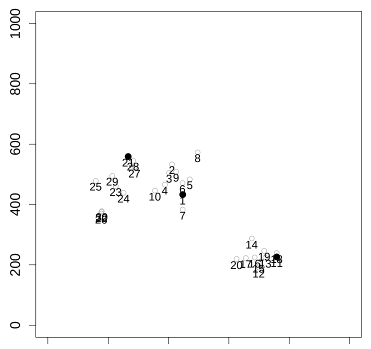
Scenario 14

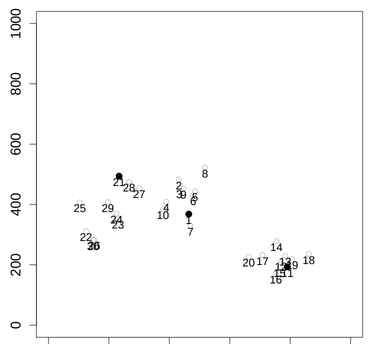

Second 0

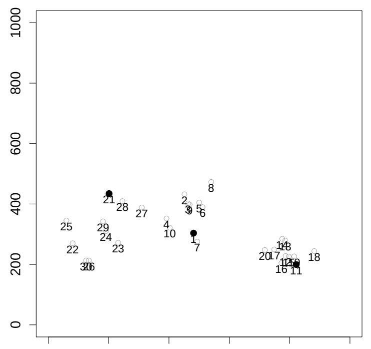

Second 100

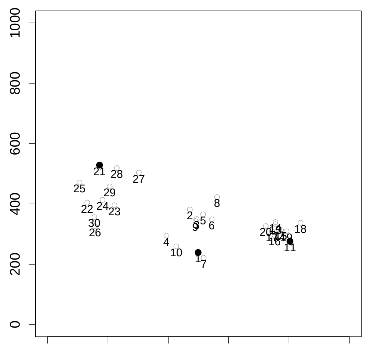

Second 200

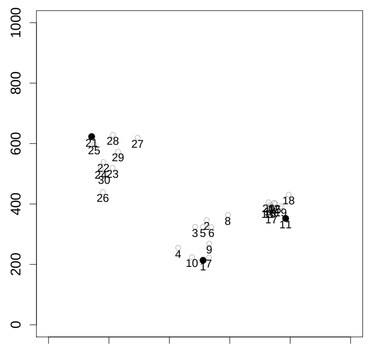

Second 300

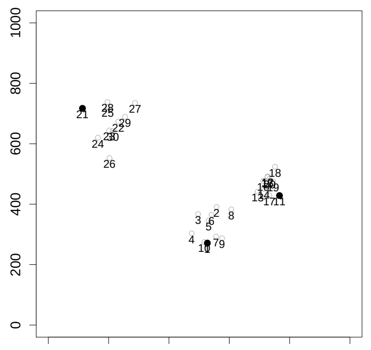

Second 400

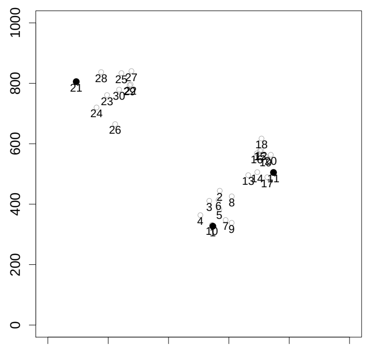

Second 500


Second 600

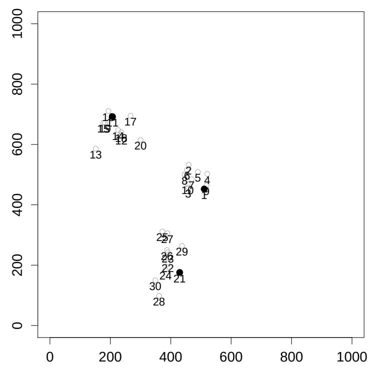

Second 700

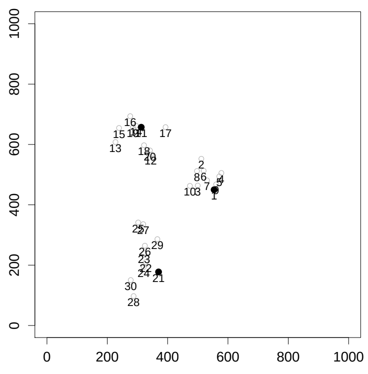

Second 800

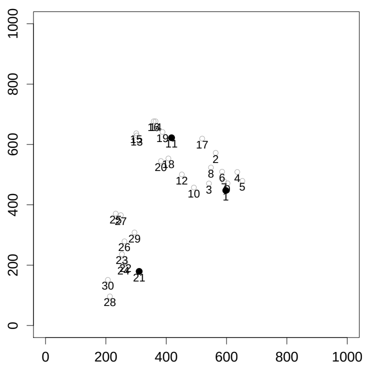

Second 900

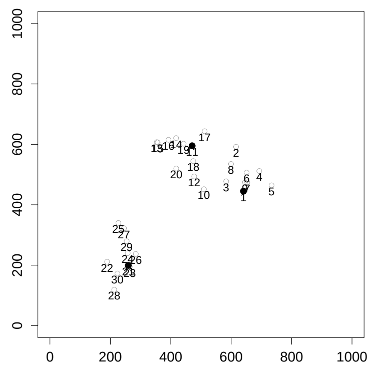

Second 1000

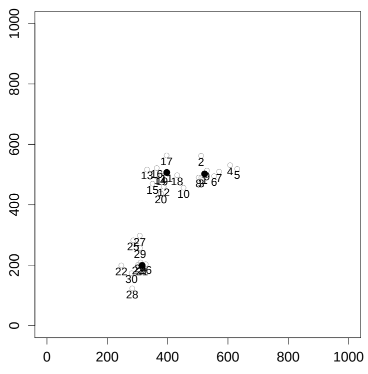
Scenario 15

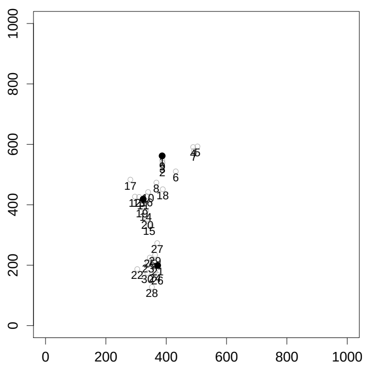

Second 0

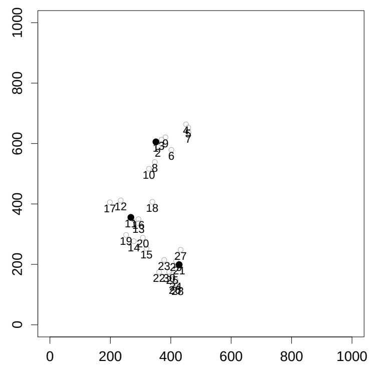

Second 100

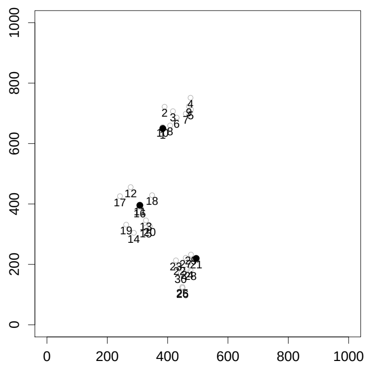

Second 200

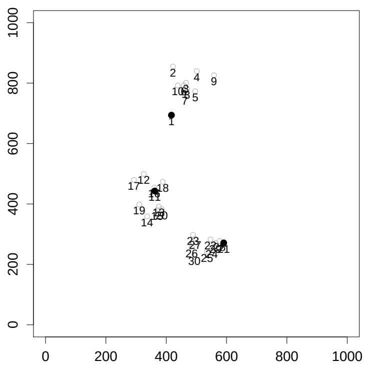

Second 300

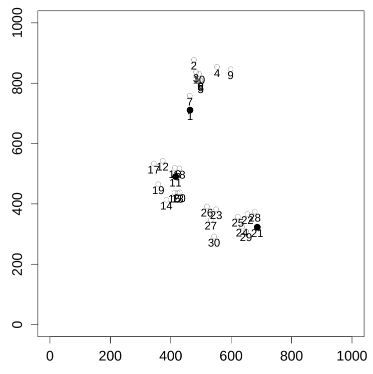

Second 400

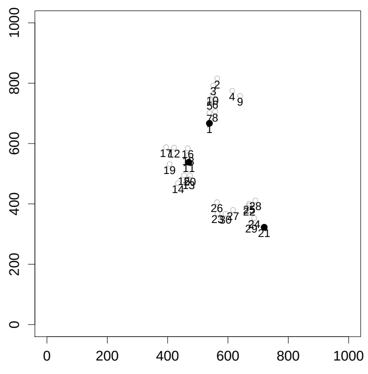

Second 500


Second 600

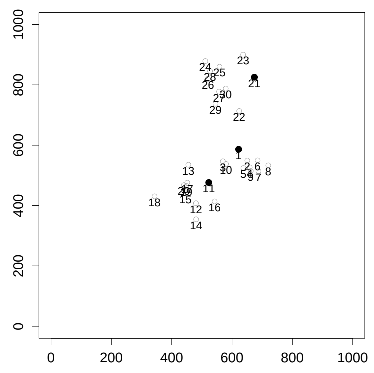

Second 700

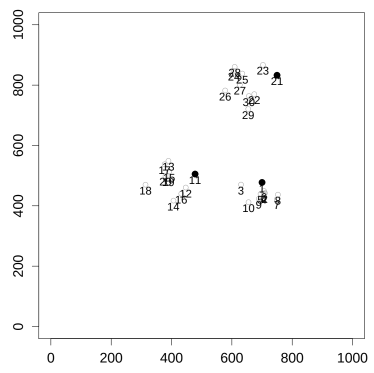

Second 800

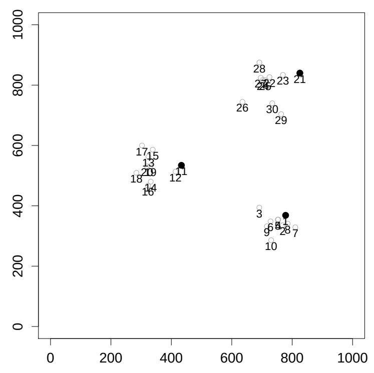

Second 900

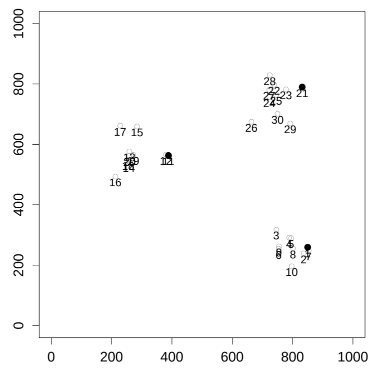

Second 1000

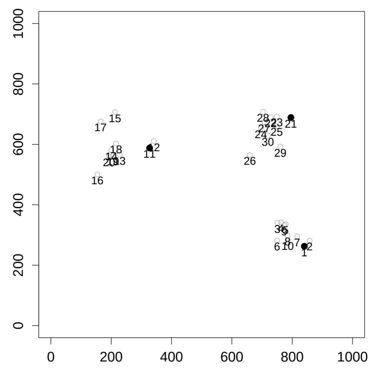
Scenario 16

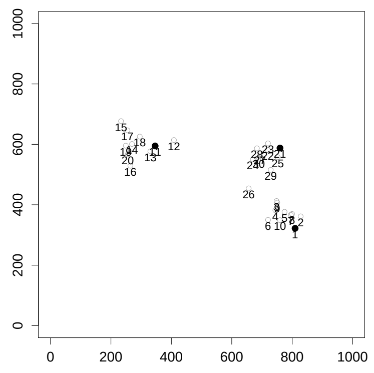

Second 0

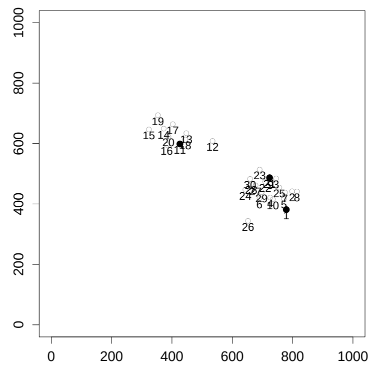

Second 100

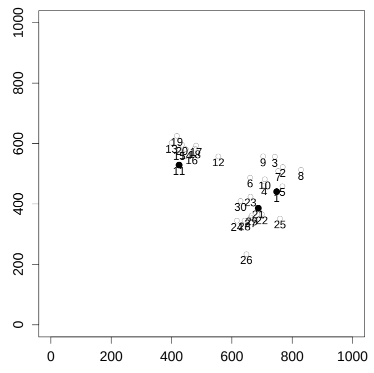

Second 200

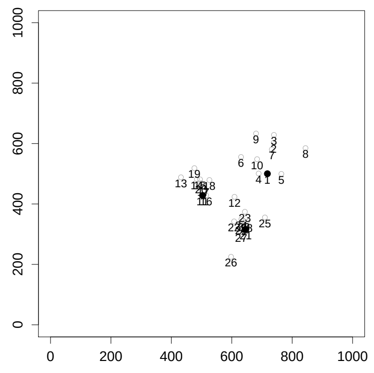

Second 300

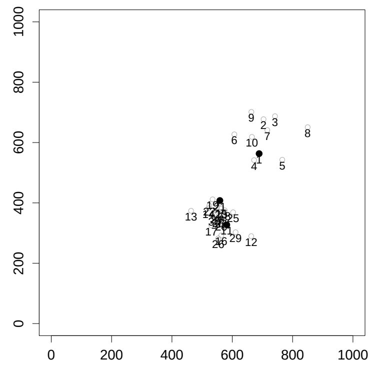

Second 400

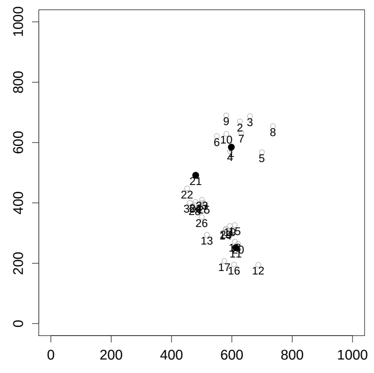

Second 500


Second 600

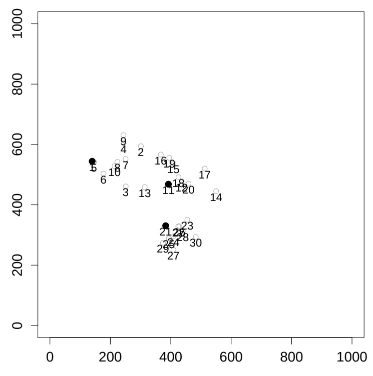

Second 700

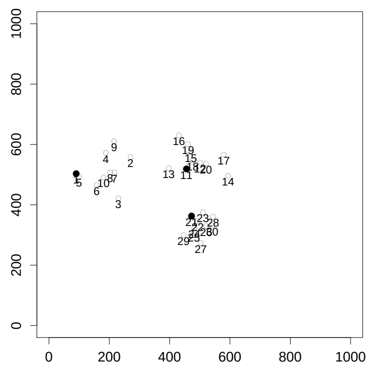

Second 800

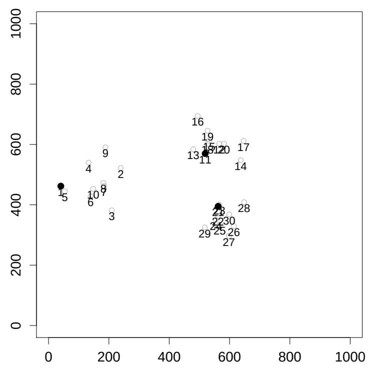

Second 900

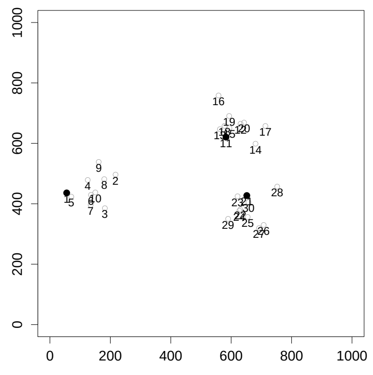

Second 1000

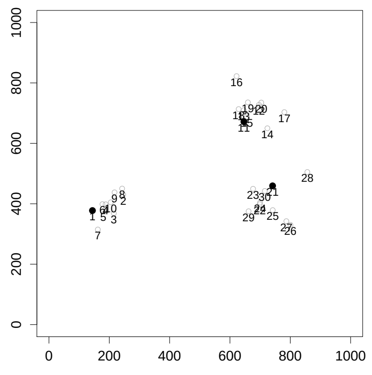
Scenario 17

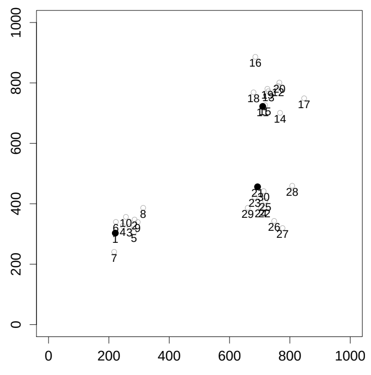

Second 0

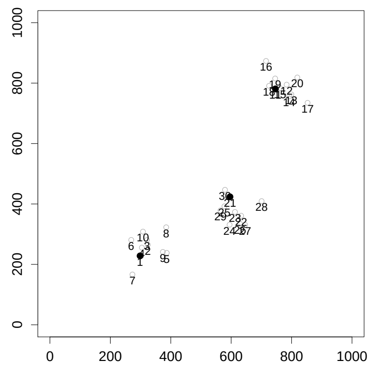

Second 100

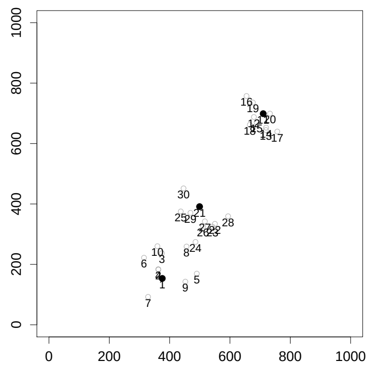

Second 200

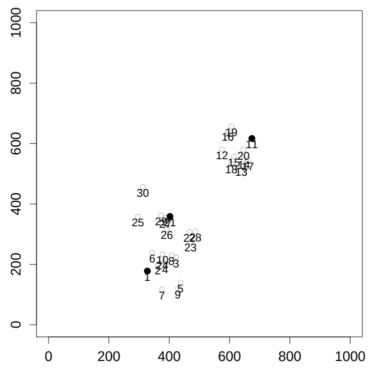

Second 300

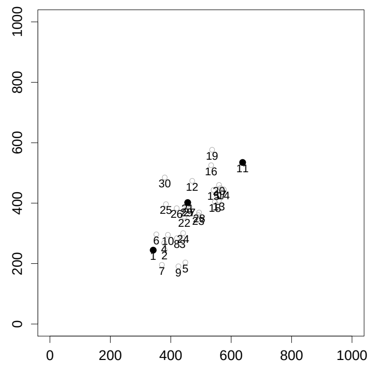

Second 400

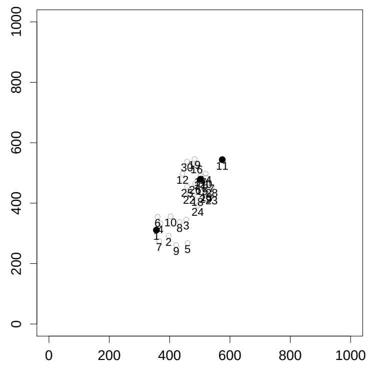

Second 500


Second 600

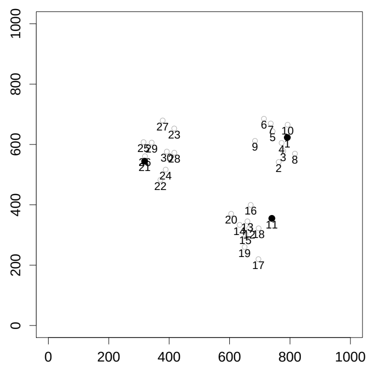

Second 700

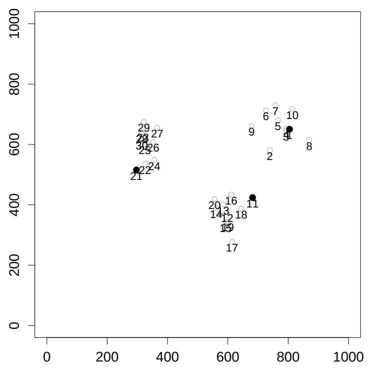

Second 800

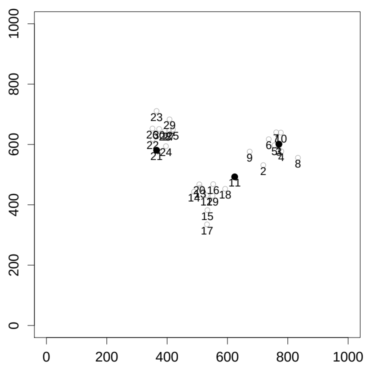

Second 900

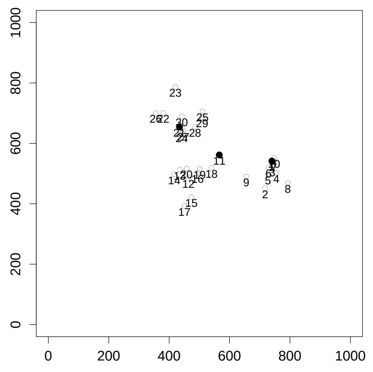

Second 1000

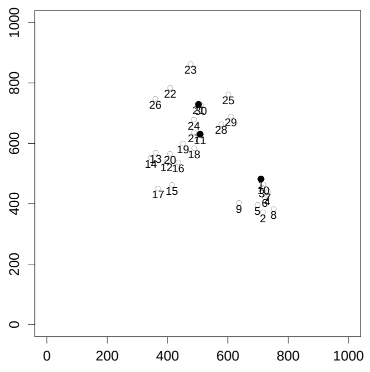
Scenario 18

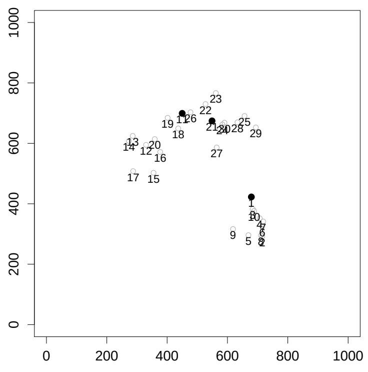

Second 0

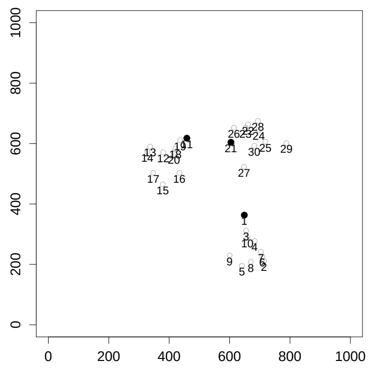

Second 100

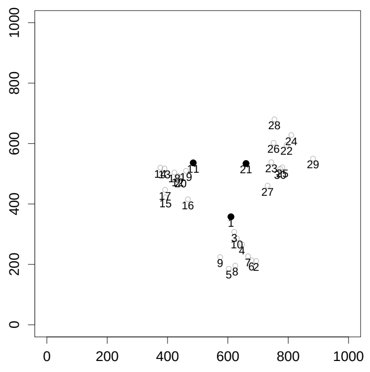

Second 200

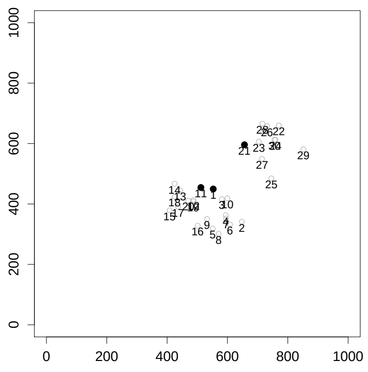

Second 300

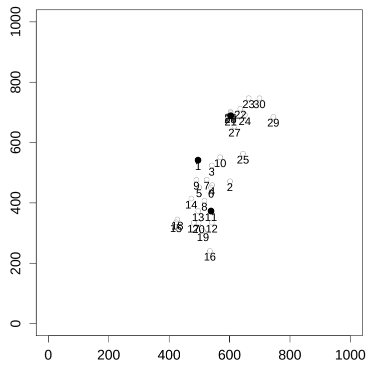

Second 400

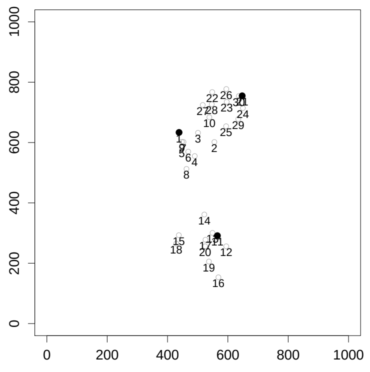

Second 500


Second 600

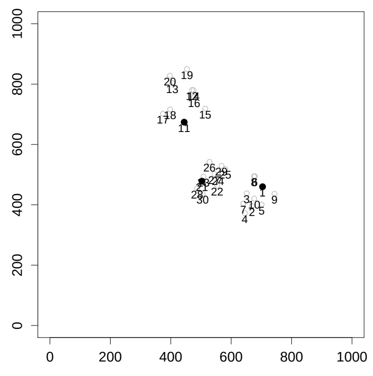

Second 700

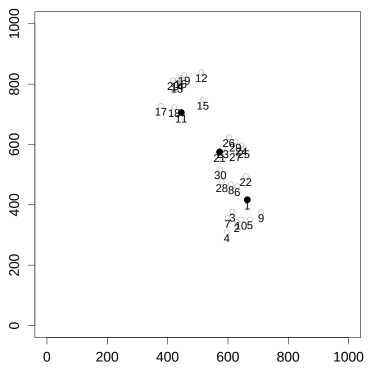

Second 800

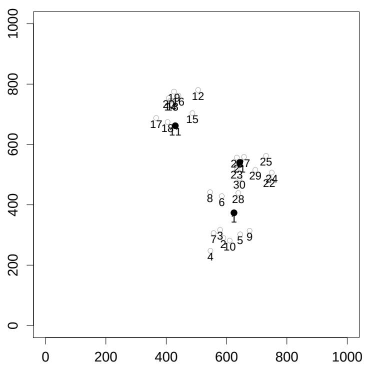

Second 900

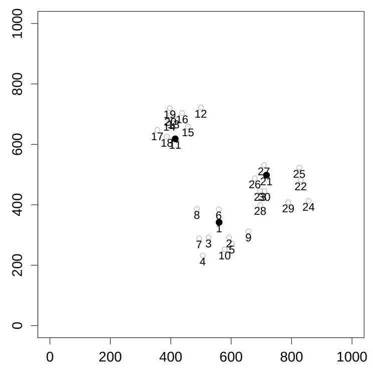

Second 1000

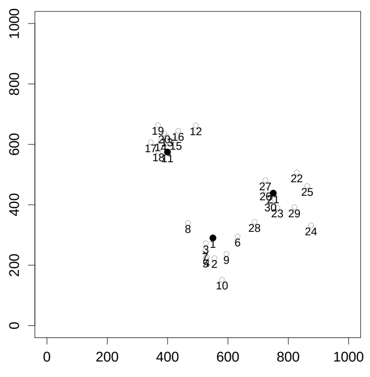
Scenario 19

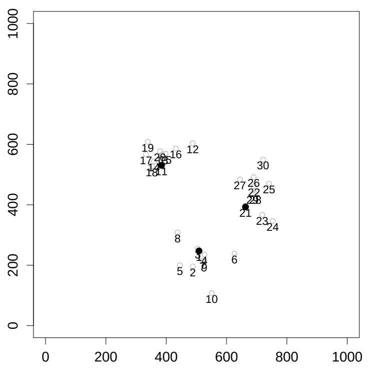

Second 0

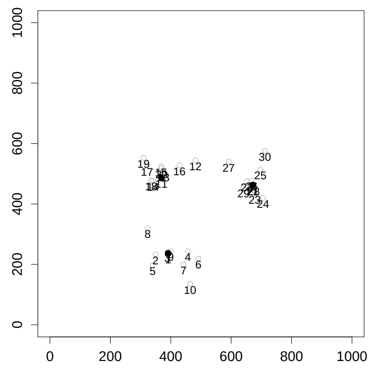

Second 100

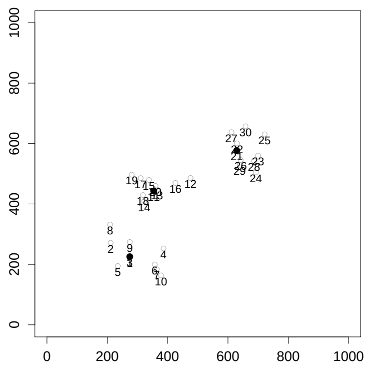

Second 200

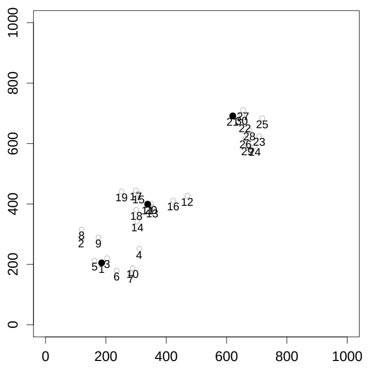

Second 300

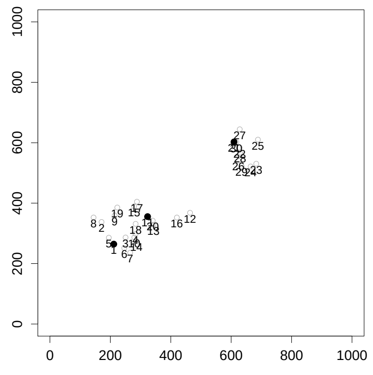

Second 400

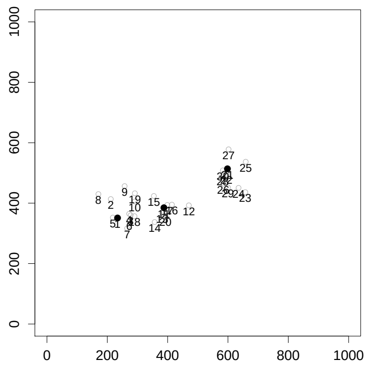

Second 500


Second 600


Second 700


Second 800

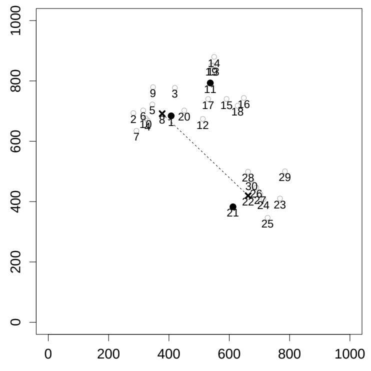

Second 900

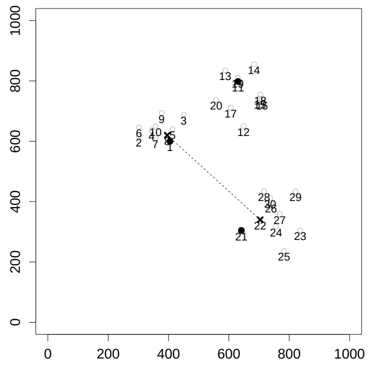

Second 1000

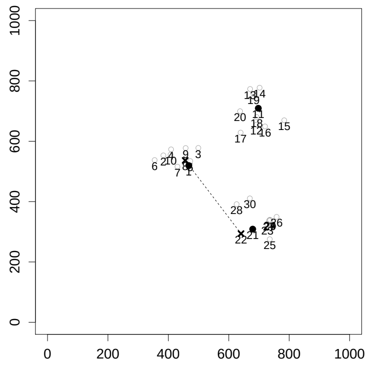
A.2. Wormhole Patterns

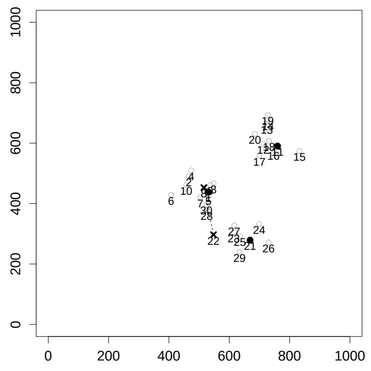
As in the previous section, these graphs display the node positions in the virtual plane, however we only included the scenarios for which we would analyse the impact of the wormhole attack. While positions are also shown in steps of 100 seconds, we limit the shown time frame to the time frame during which the attack took place. The wormhole endpoints are printed as black crosses, connected by a dashed line, nodes participating in the command and control communication group are printed as solid black dots while all other nodes are printed as grey circles. Both x and y axis reflect the respective position in meters.

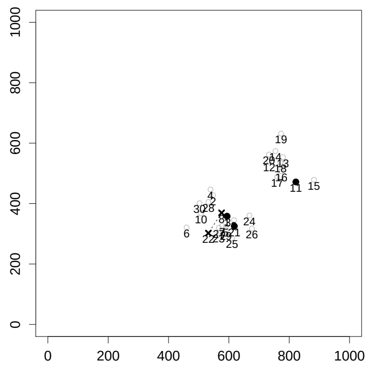
Scenario 0


Second 600

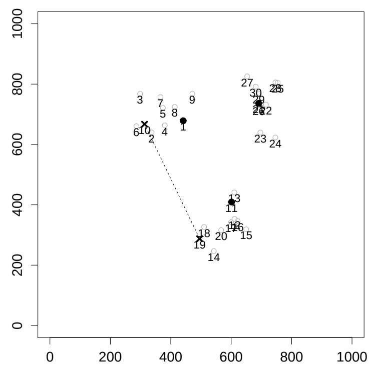

Second 700

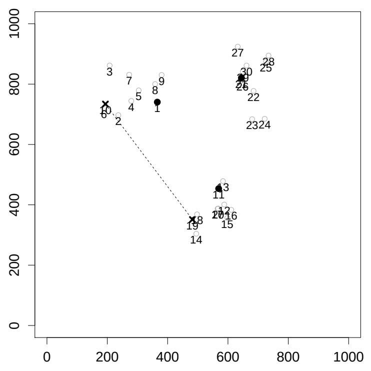

Second 800

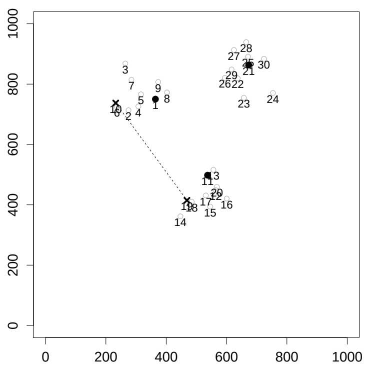

Second 900

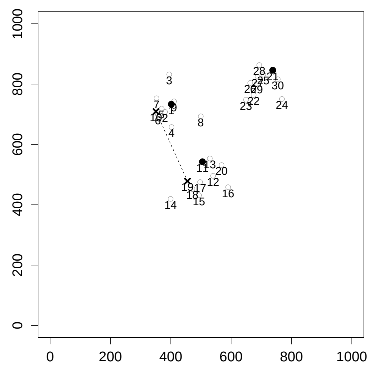

Second 1000

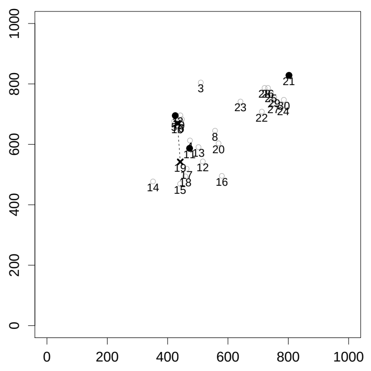
Scenario 1


Second 600

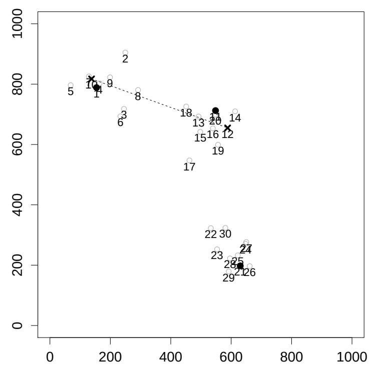

Second 700

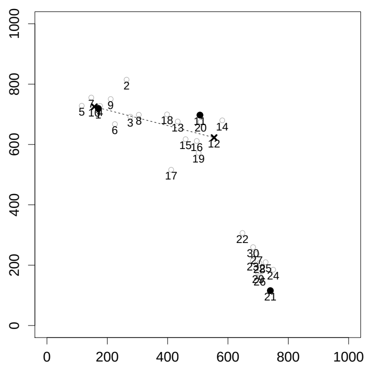

Second 800

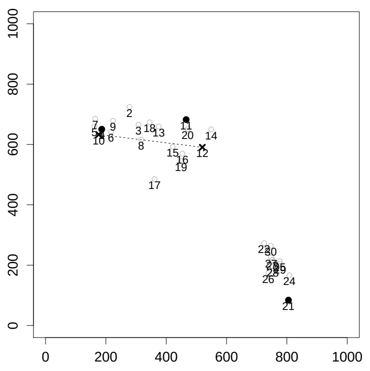

Second 900

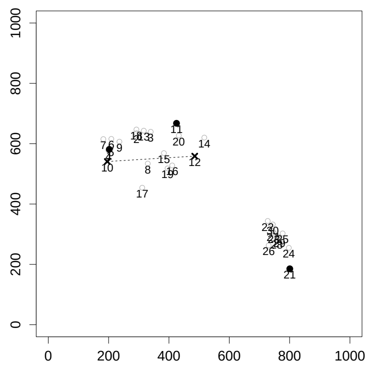

Second 1000

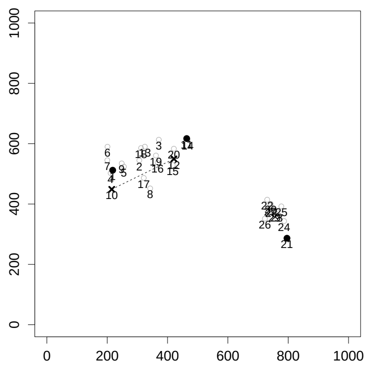
Scenario 2


Second 600

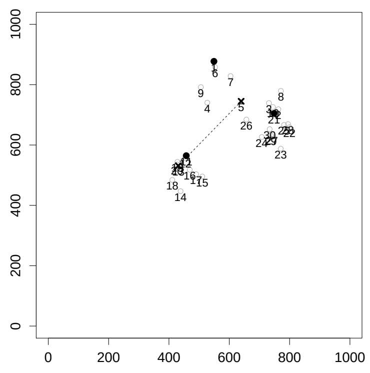

Second 700

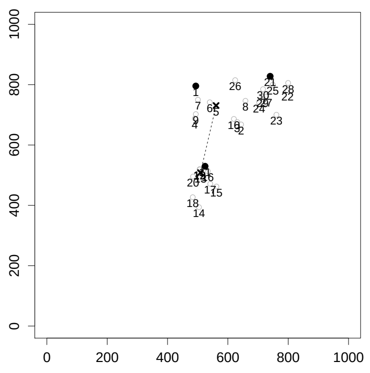

Second 800

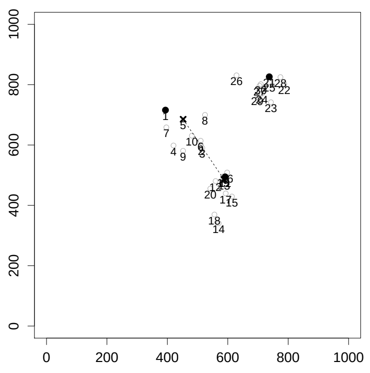

Second 900

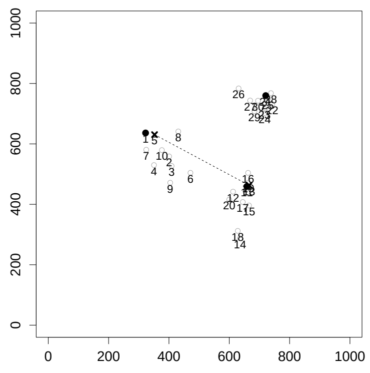

Second 1000

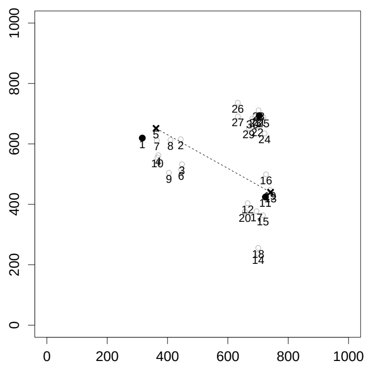
Scenario 3


Second 600

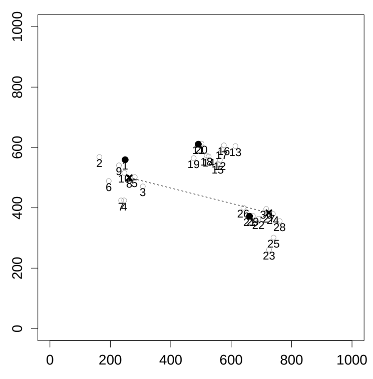

Second 700

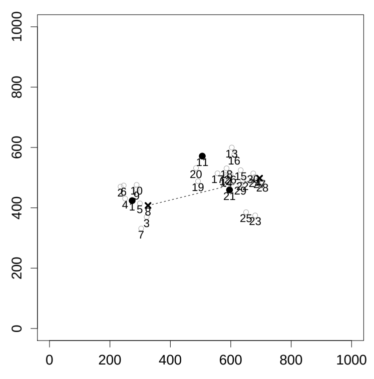

Second 800

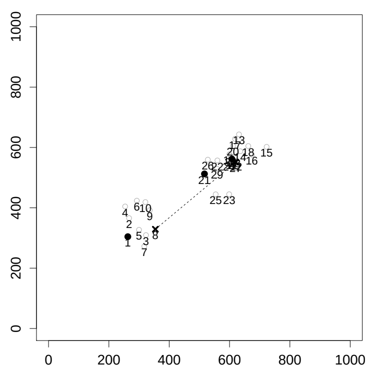

Second 900

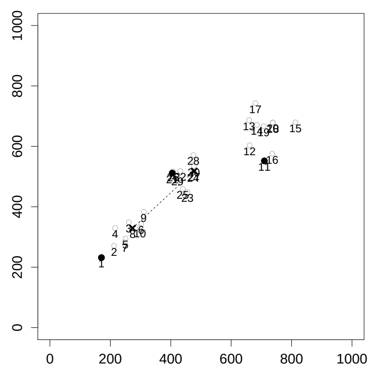

Second 1000

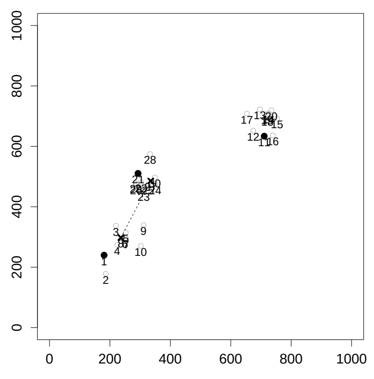
Scenario 4


Second 600

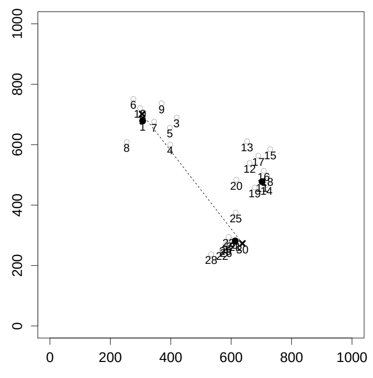

Second 700

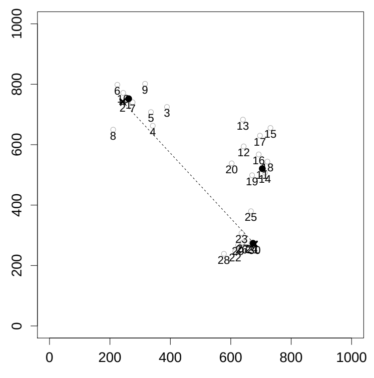

Second 800

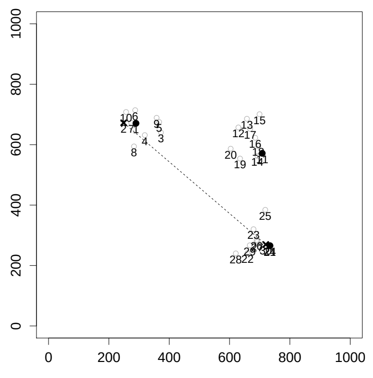

Second 900

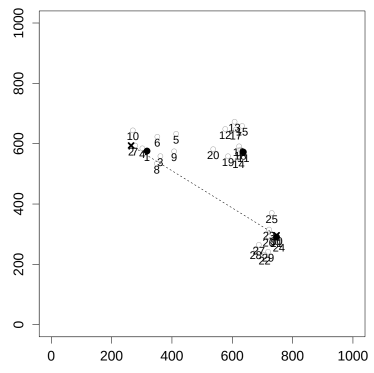

Second 1000

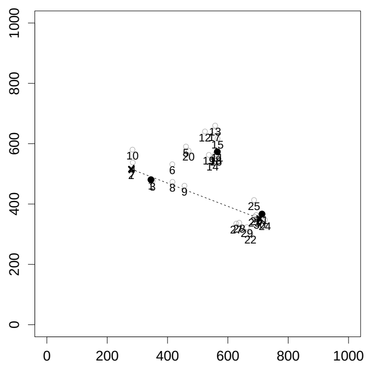
Scenario 5


Second 600

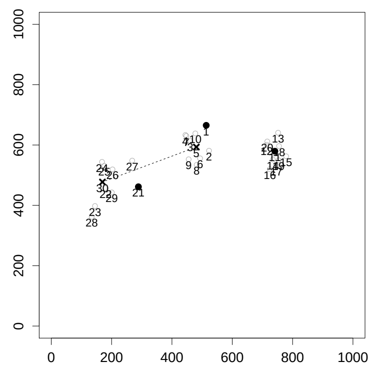

Second 700

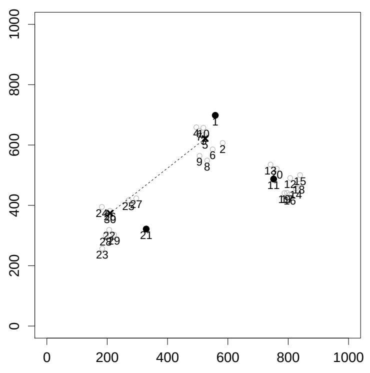

Second 800

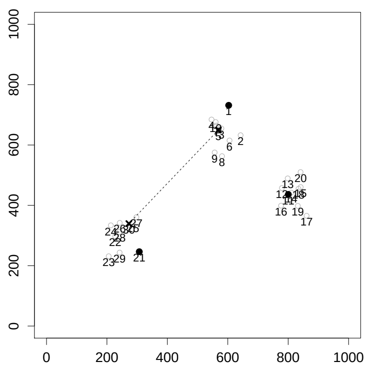

Second 900

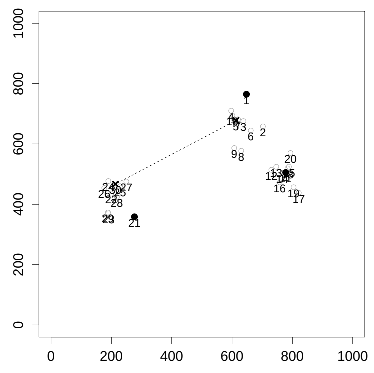

Second 1000

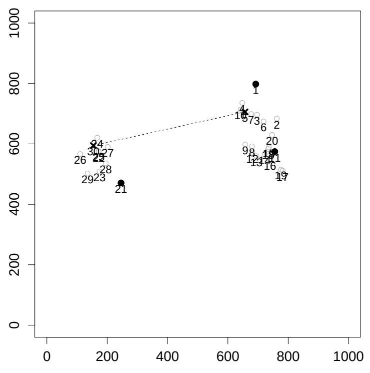
Scenario 7


Second 600

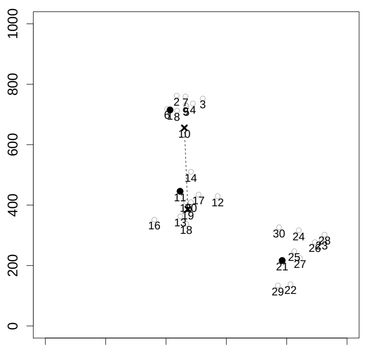

Second 700

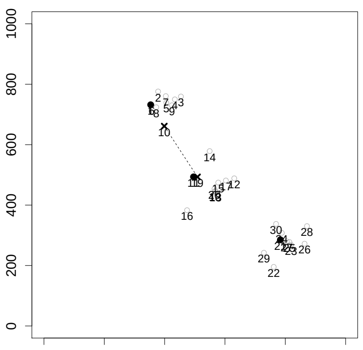

Second 800

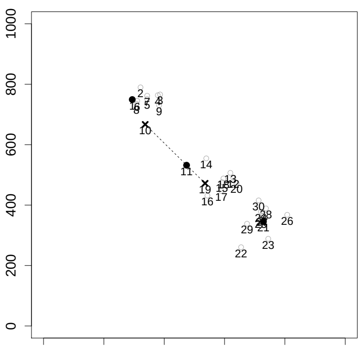

Second 900

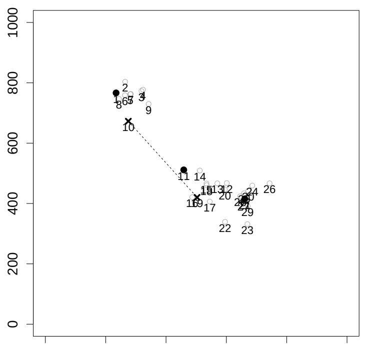

Second 1000

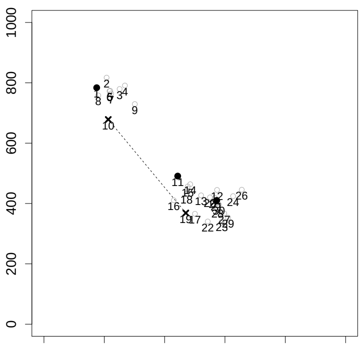
Scenario 8


Second 600

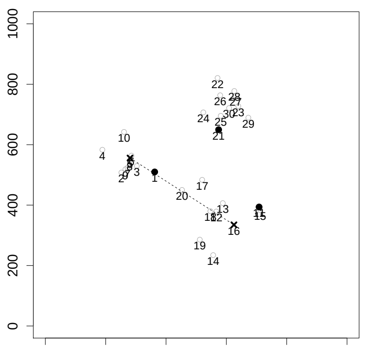

Second 700

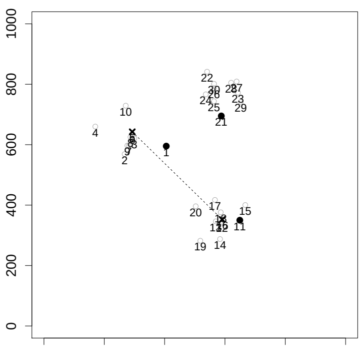

Second 800

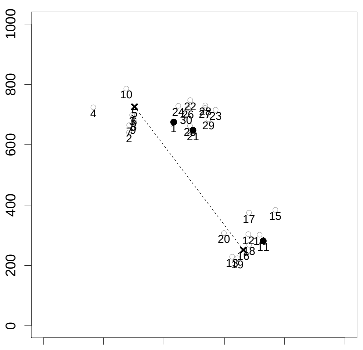

Second 900

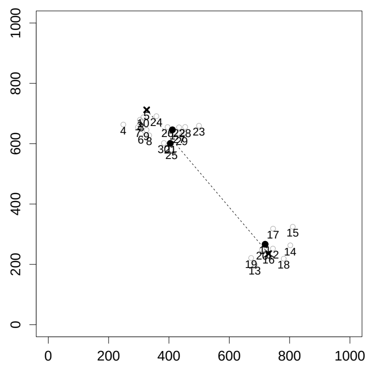

Second 1000

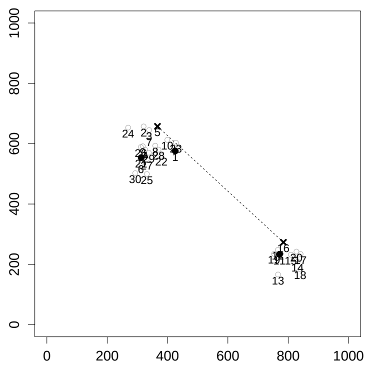
Scenario 9


Second 600

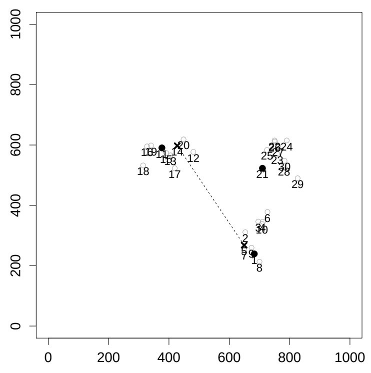

Second 700

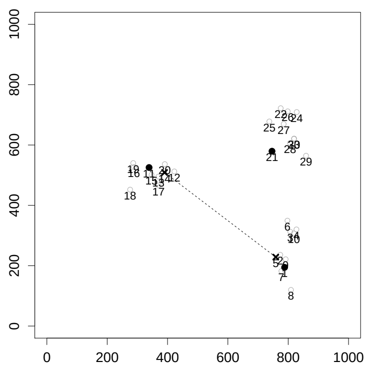

Second 800

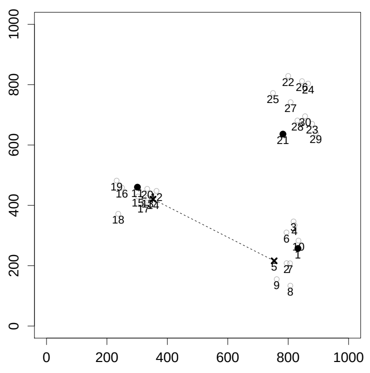

Second 900

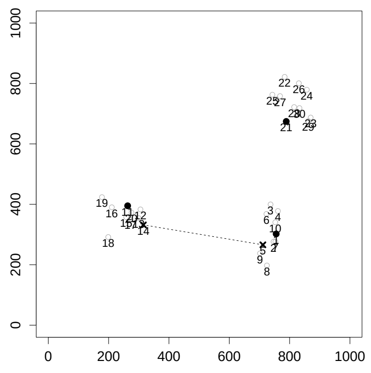

Second 1000

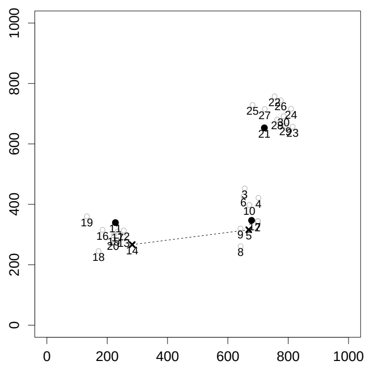
Scenario 10


Second 600

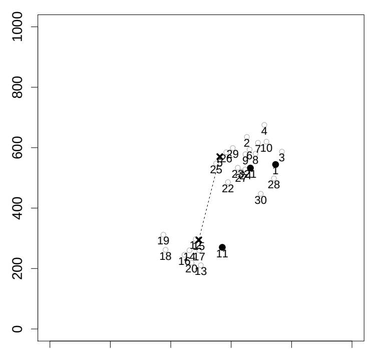

Second 700

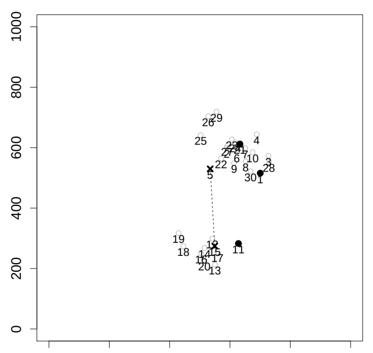

Second 800

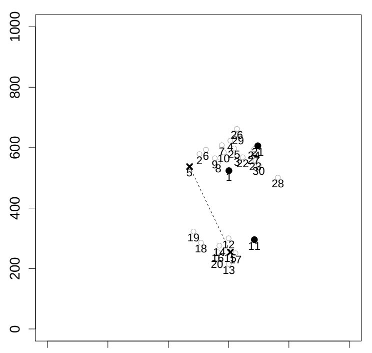

Second 900

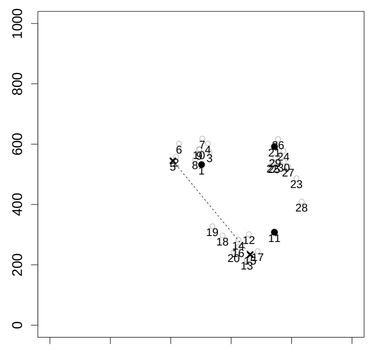

Second 1000

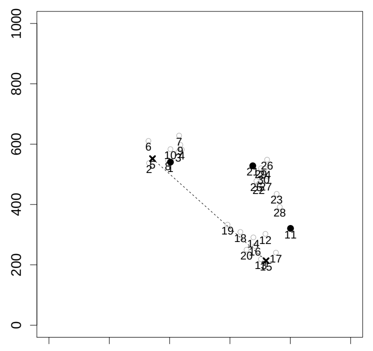
Scenario 11


Second 600

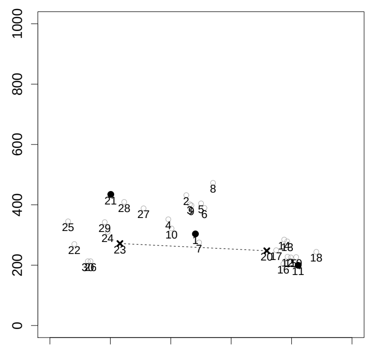

Second 700

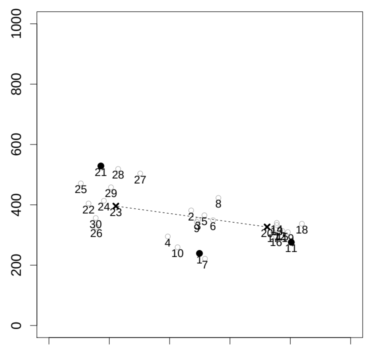

Second 800

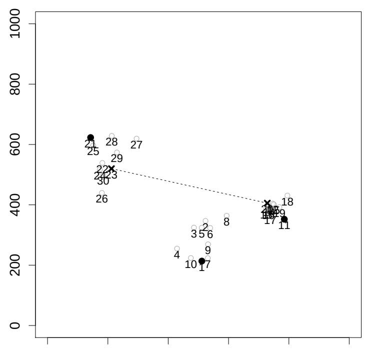

Second 900

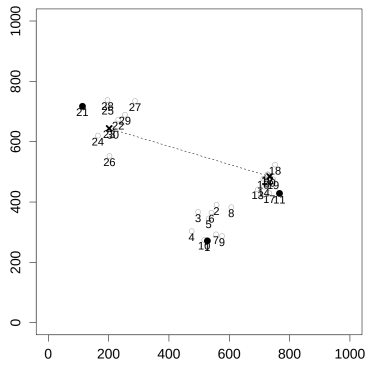

Second 1000

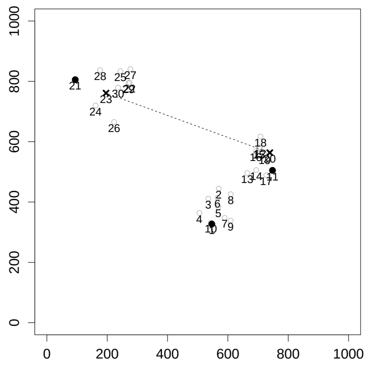
Scenario 14


Second 600

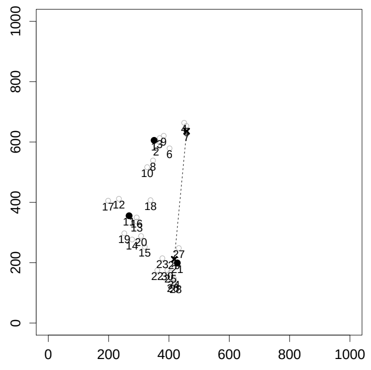

Second 700

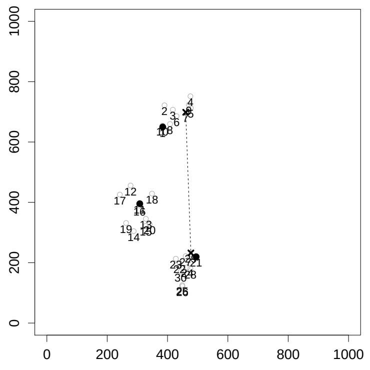

Second 800

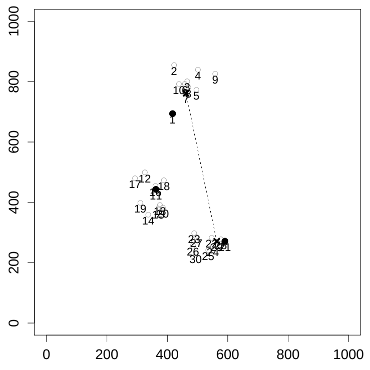

Second 900

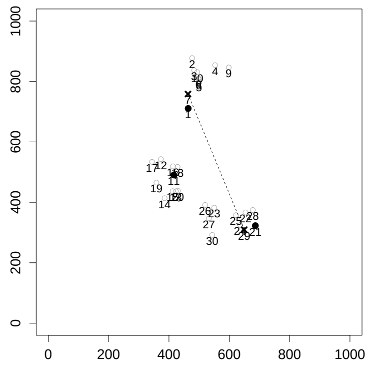

Second 1000

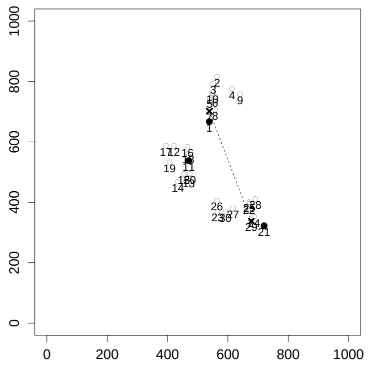
Scenario 15


Second 600

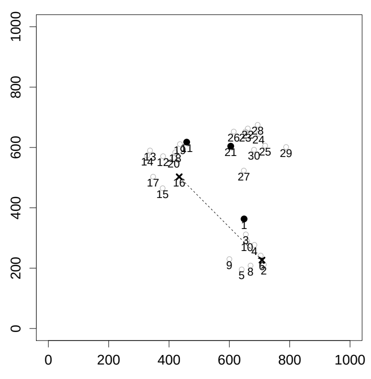

Second 700

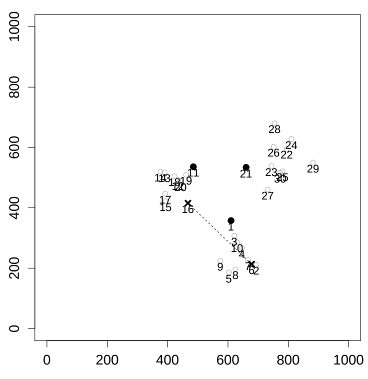

Second 800

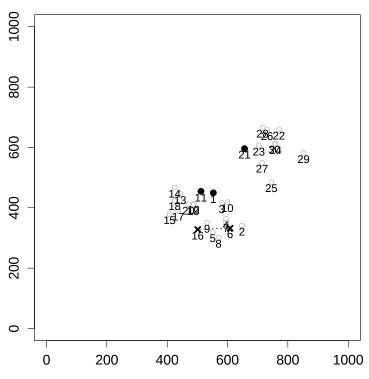

Second 900

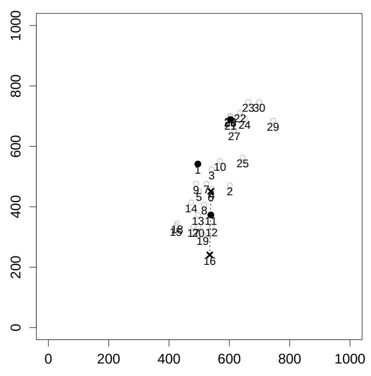

Second 1000

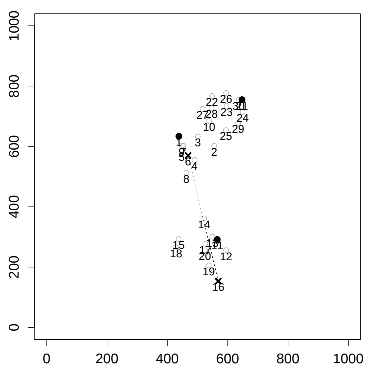
Scenario 18


Second 600

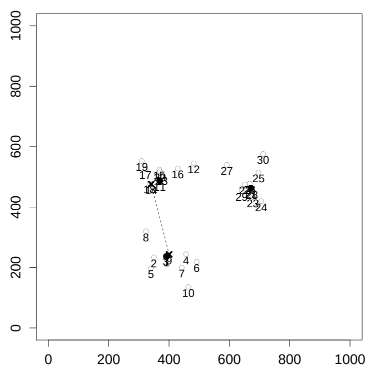

Second 700

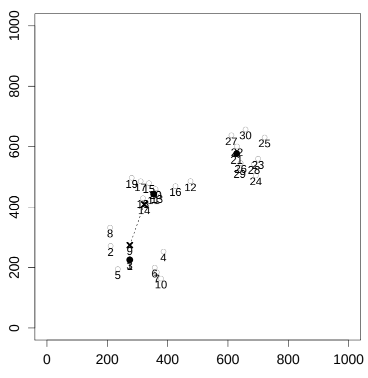

Second 800

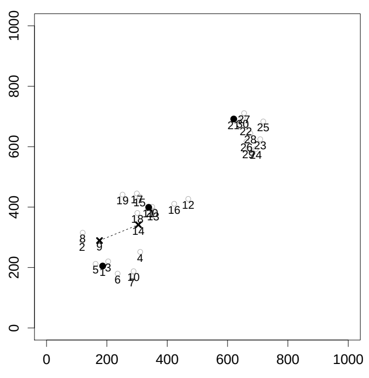

Second 900

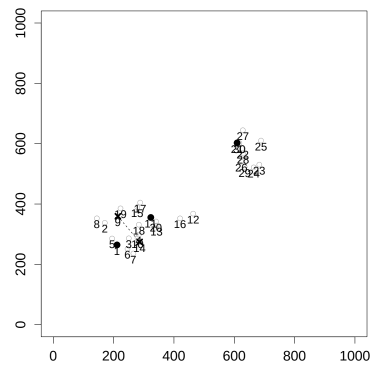

Second 1000

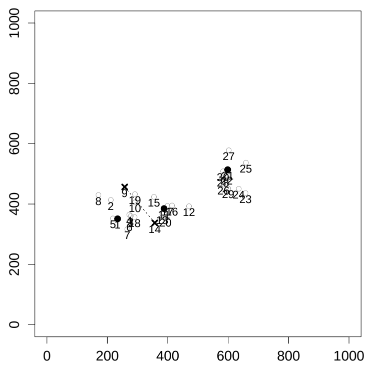
Scenario 19


Second 600


Second 700


Second 800


Second 900


Second 1000

Appendix B

Packet Delivery Ratios in Selected Emulation Results

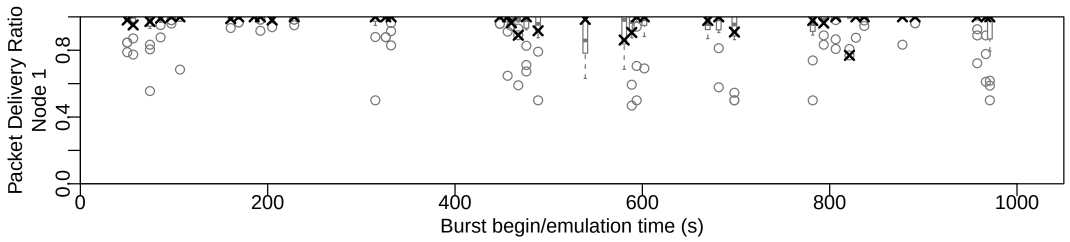
This chapter provides an overview to the PDR obtained for each individual burst of packets in 15 emulations executed with the same configuration and for each individual scenario and the PDR in the scenario selected based on the criterion described in Section 5.3.

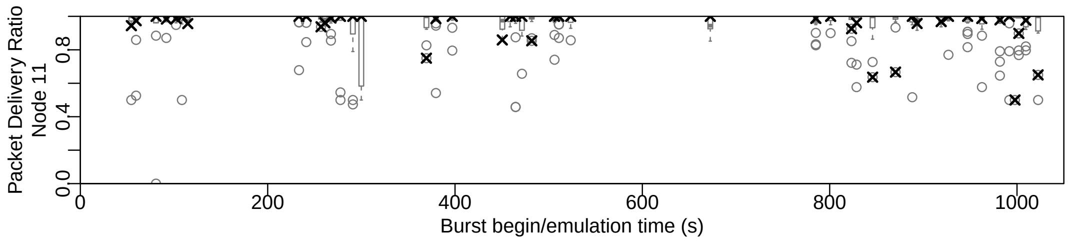
Note that, as pointed out in the named section, each of the emulations considered here passed the CPU criterion, i.e. none of the computers involved in them experienced an exhaustion of its CPU resources for 2 or more seconds. In these graphs, each box represents the PDRs obtained in a given burst of packets in each of the named emulations where the position on the x-axis refers to the emulation timestamp at which the burst started while the ordinate reflects the actual PDR achieved. Note that we had to adjust the width of the boxes to avoid too many overlaps while trying to maintaining readability. In some cases, it was impossible to achieve either of these goals.

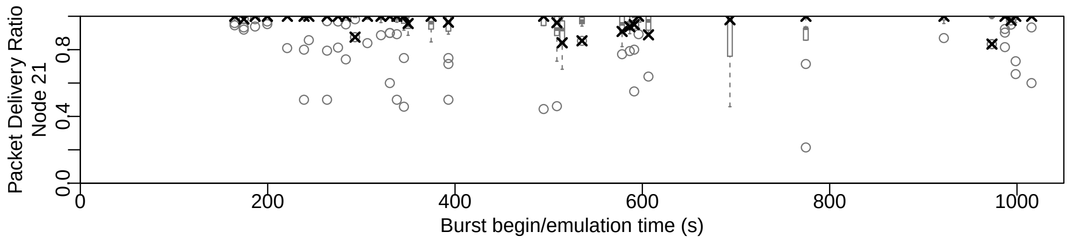
In addition to the boxes, each graph contains black crosses reflecting the PDR achieved in the selected emulation results for the given scenario. As described in Section 5.3, we were not able to obtain the desired number of results for all scenarios when executing emulations with attacks, thus, while Section B.1 presents results for each scenario, Section B.2 only contains graphs for those scenarios where our required count of results was reached.

B.1. Packet Delivery Ratios in Emulations Without Attacks

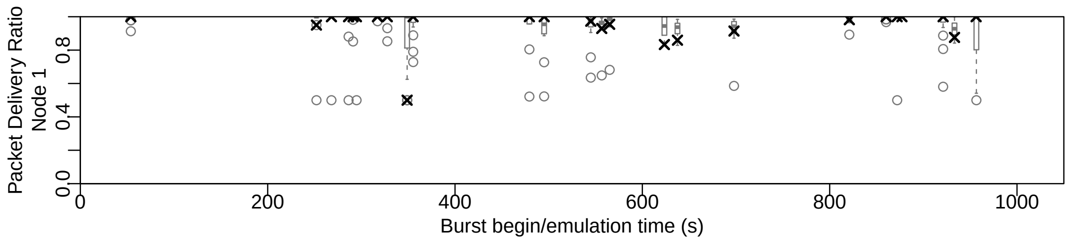
Scenario 0

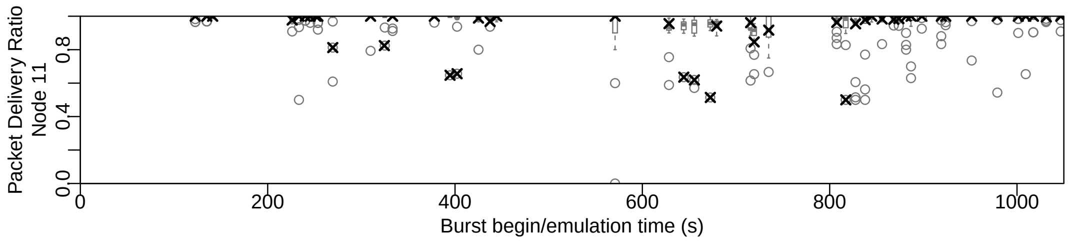


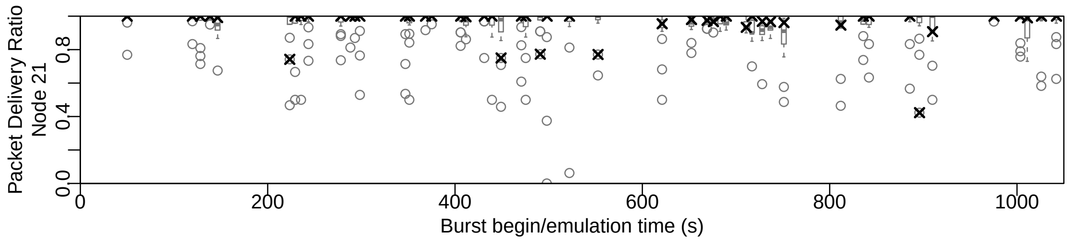




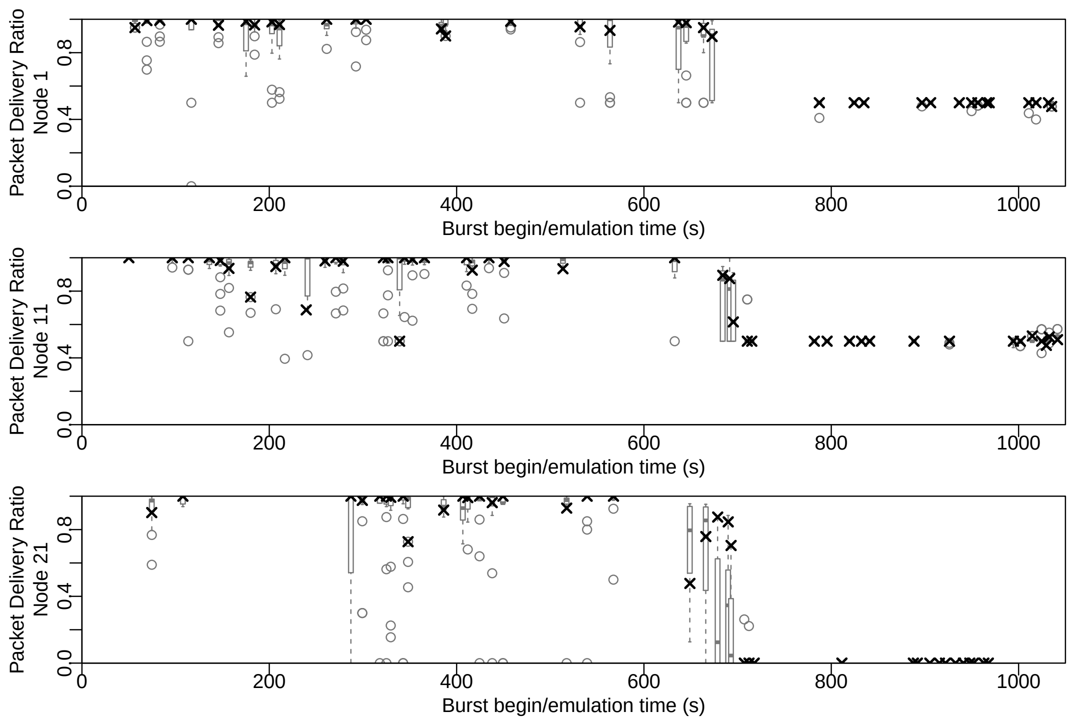
Scenario 1



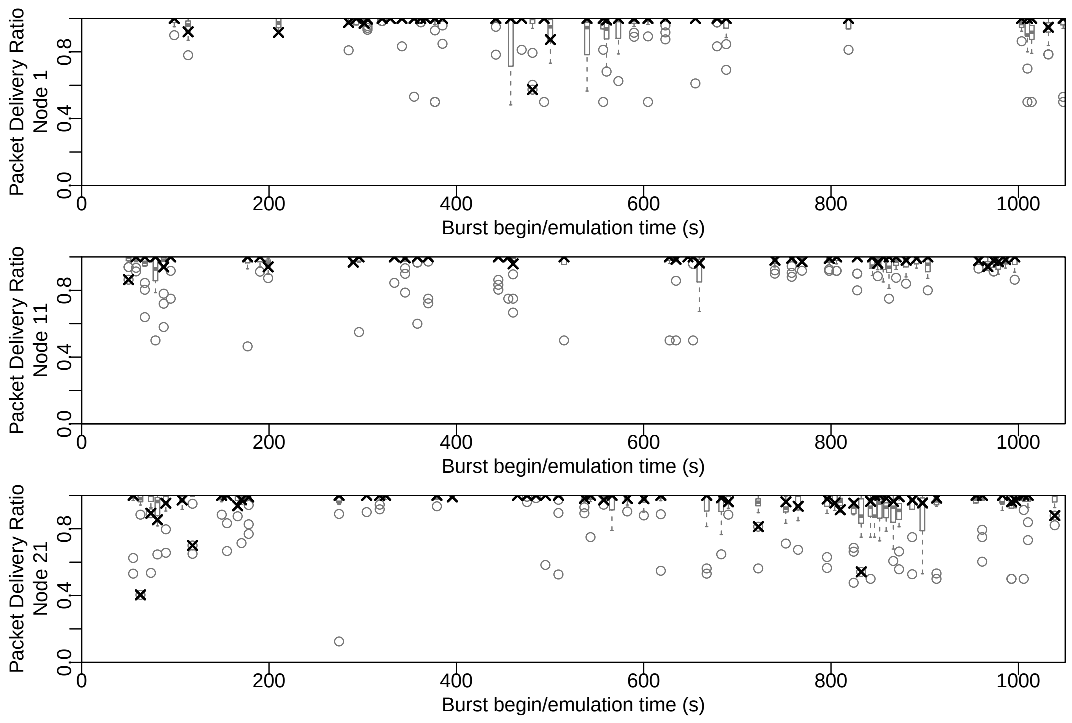




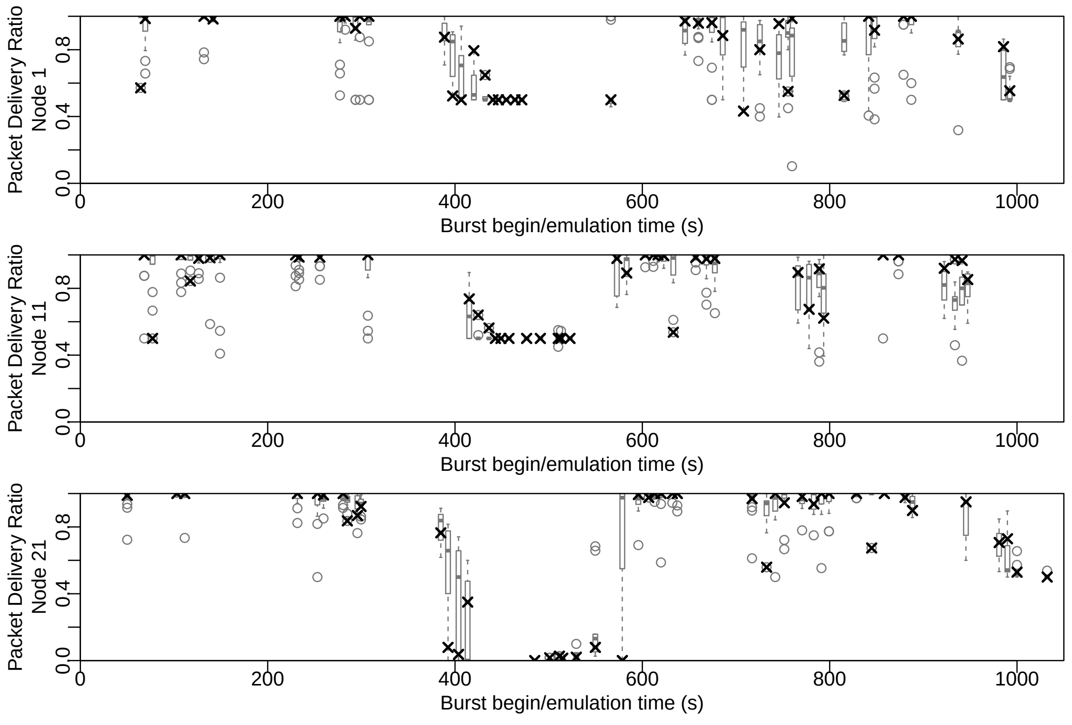
Scenario 2



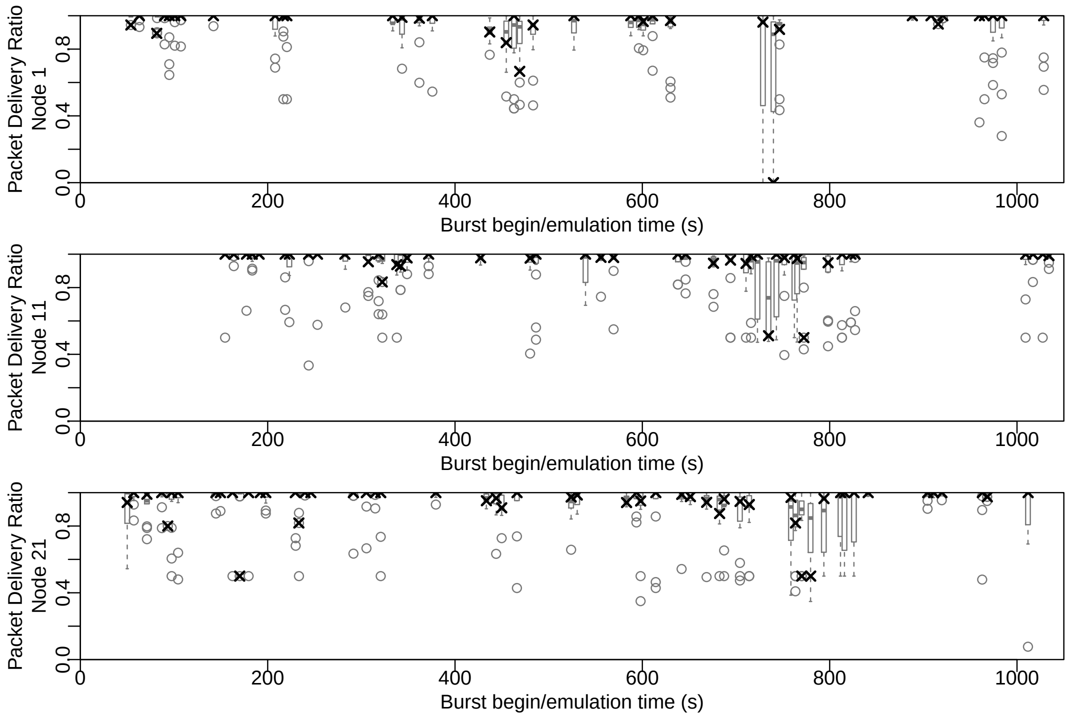
Scenario 3



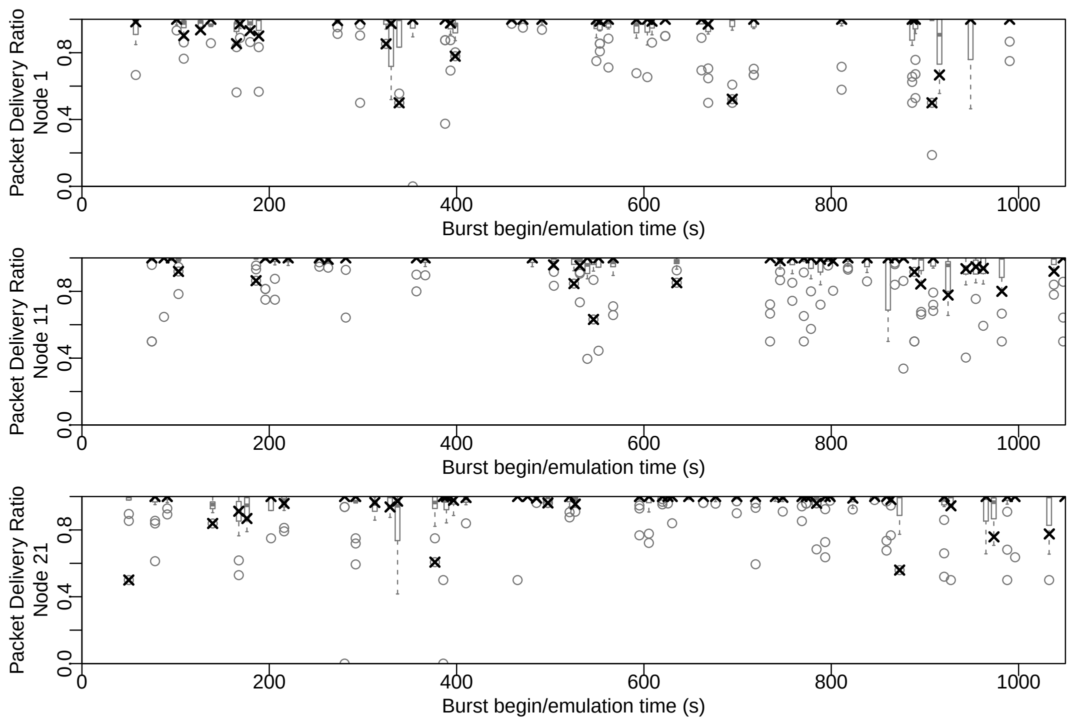
Scenario 4



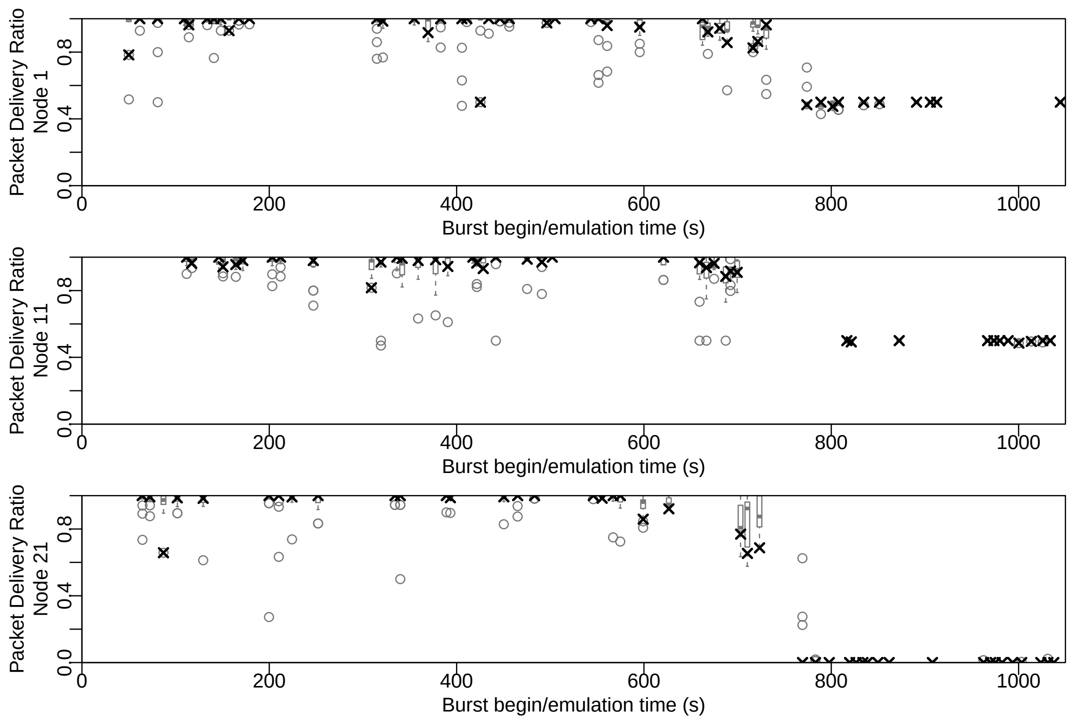
Scenario 5



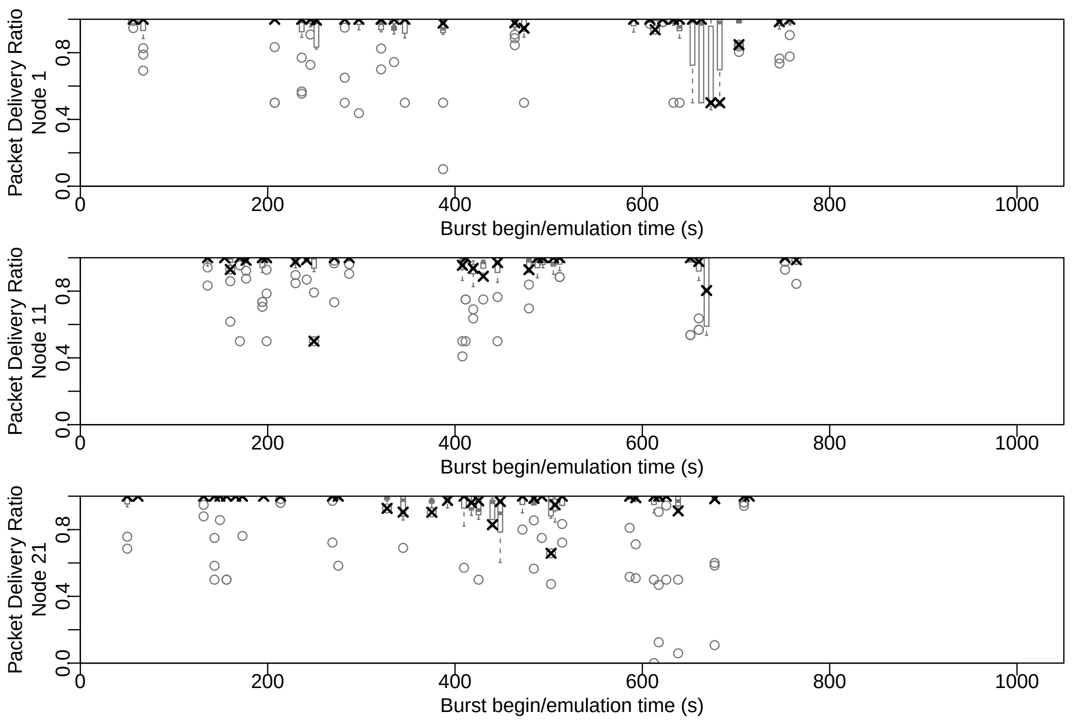
Scenario 6



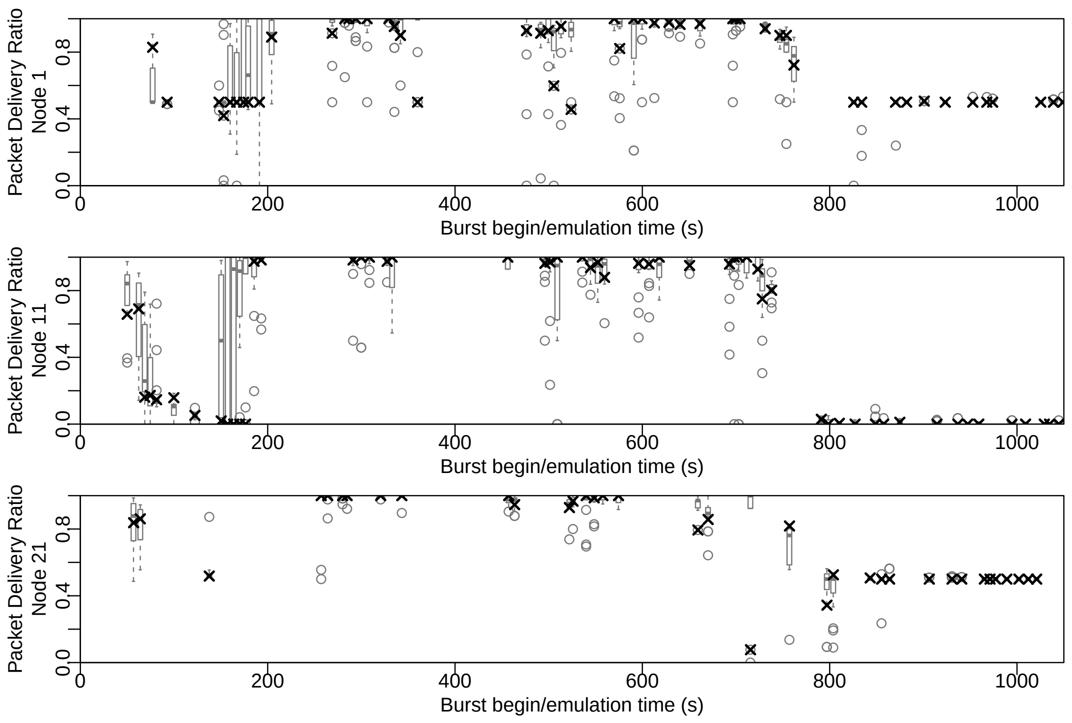
Scenario 7



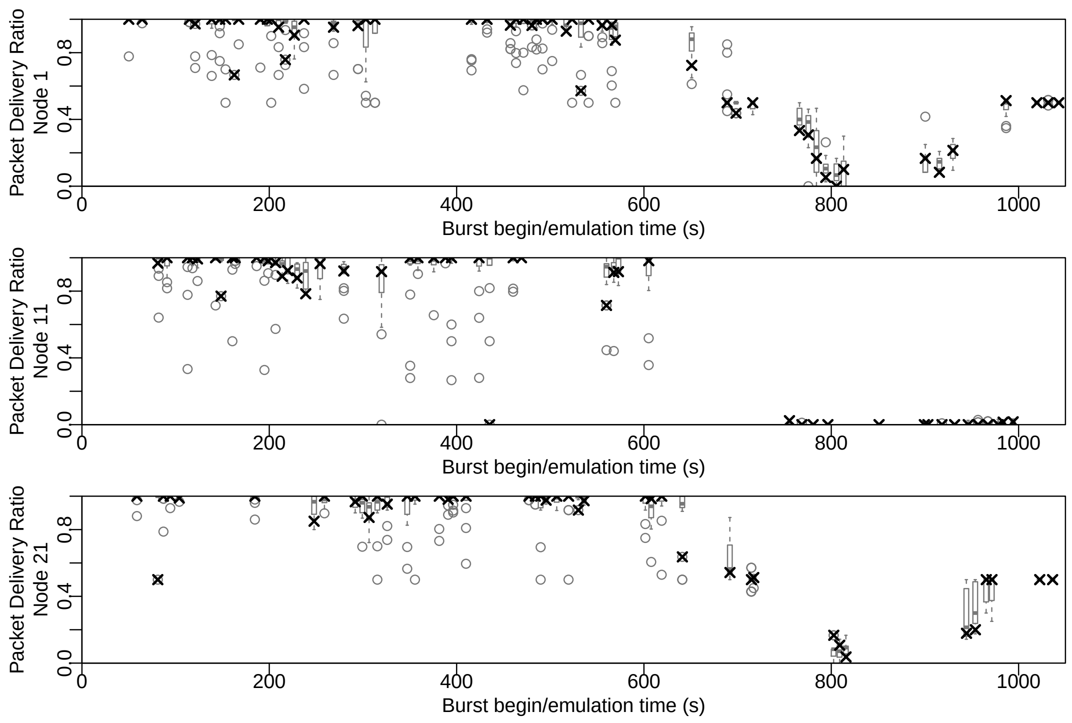
Scenario 8



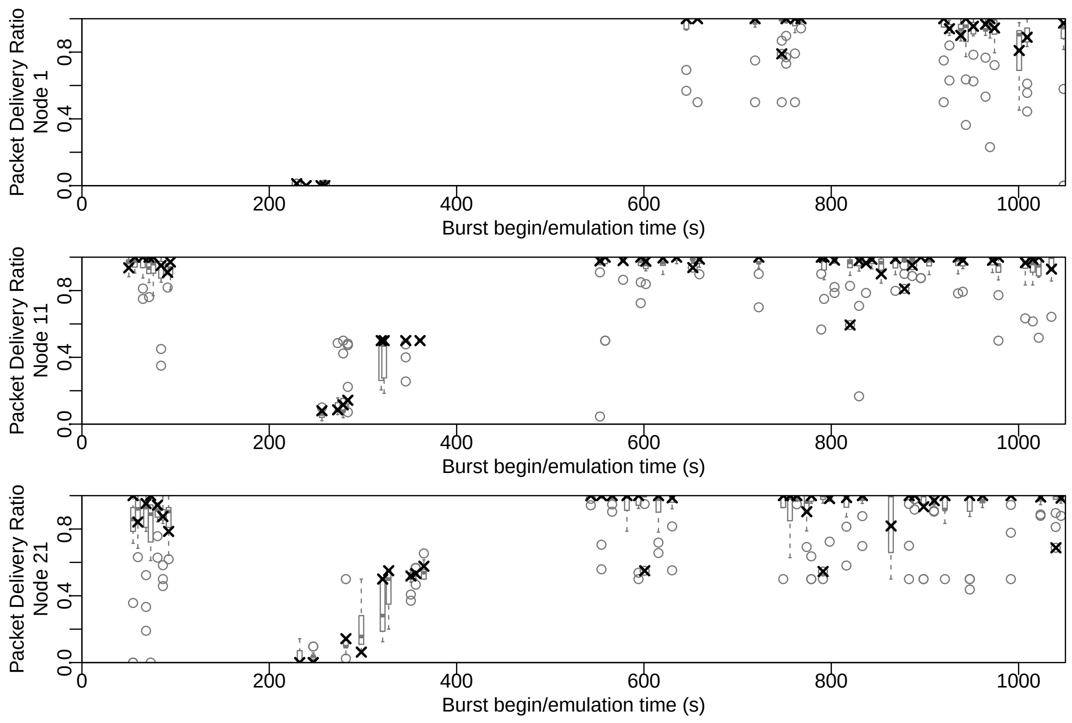
Scenario 9



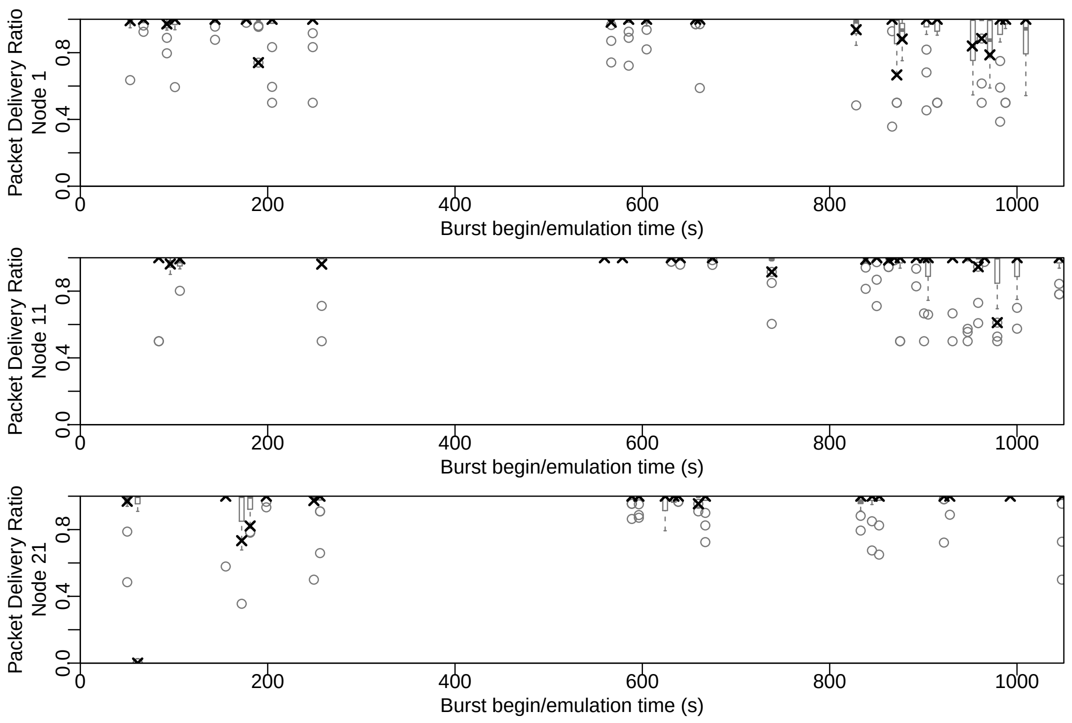
Scenario 10



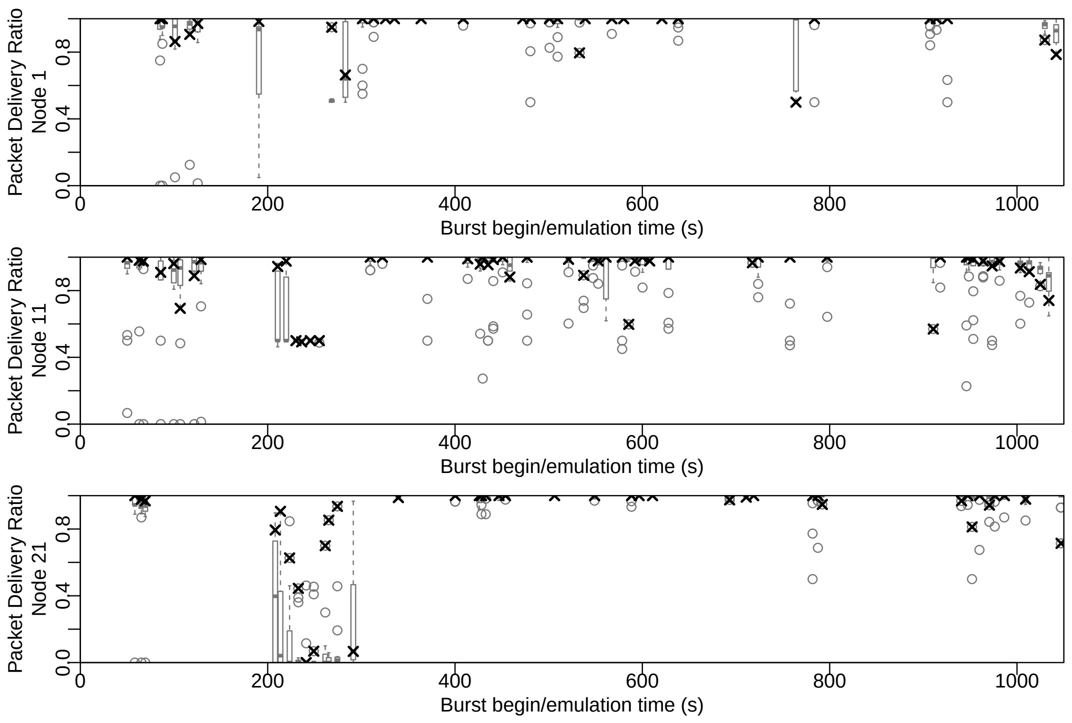
Scenario 11



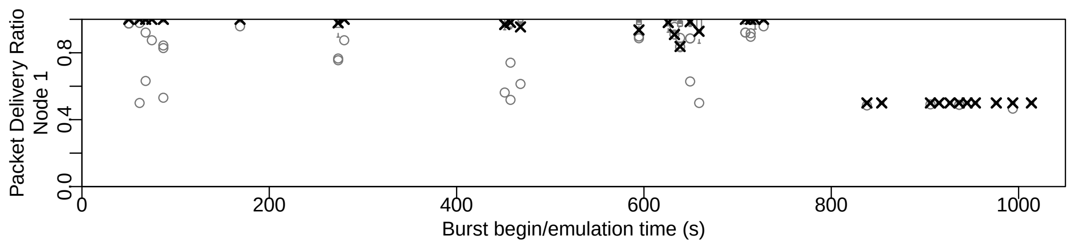
Scenario 12

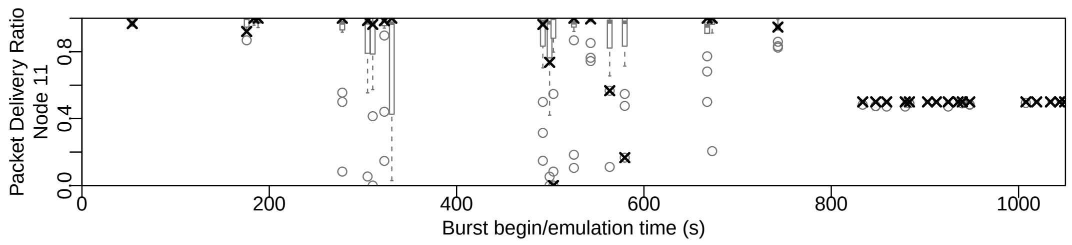


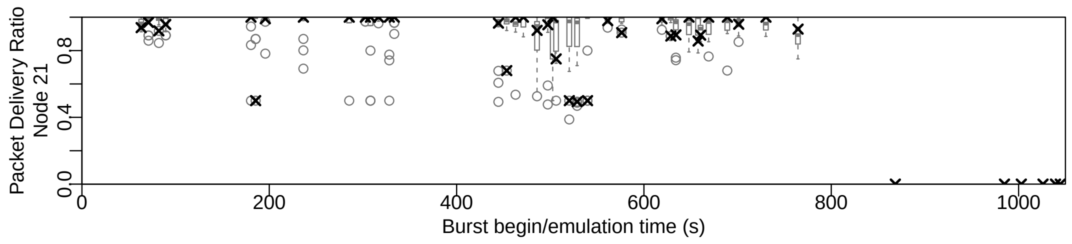
Scenario 13



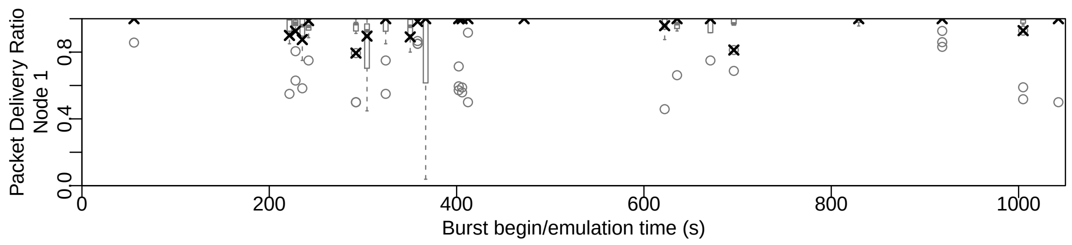
Scenario 14

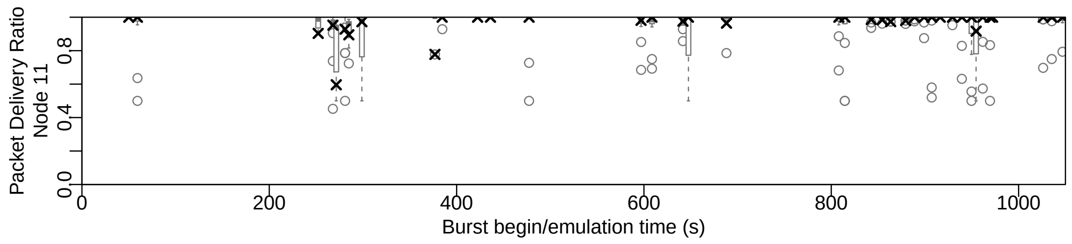


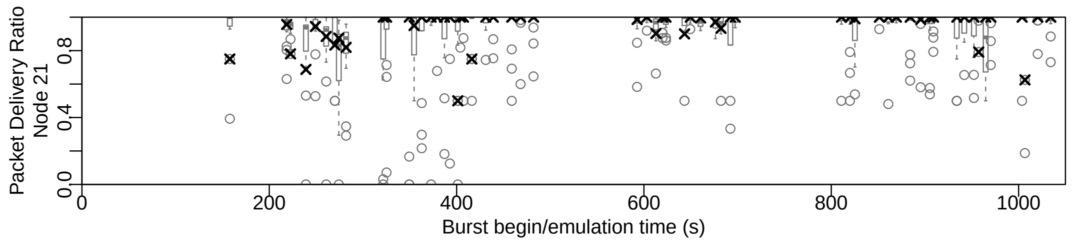




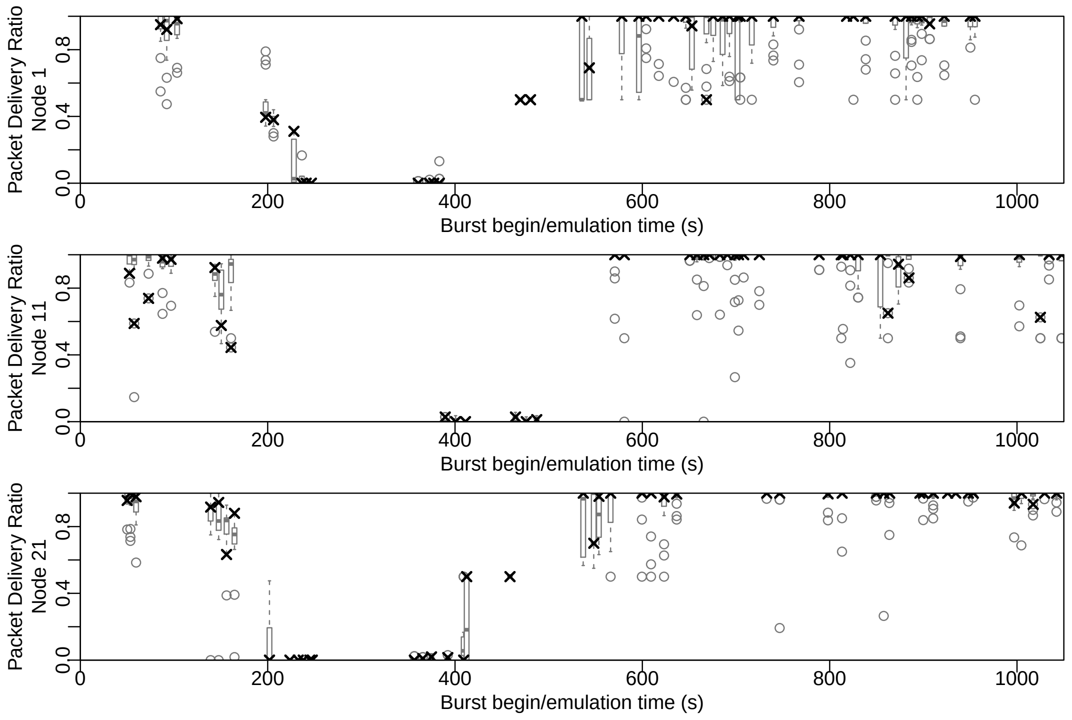
Scenario 15



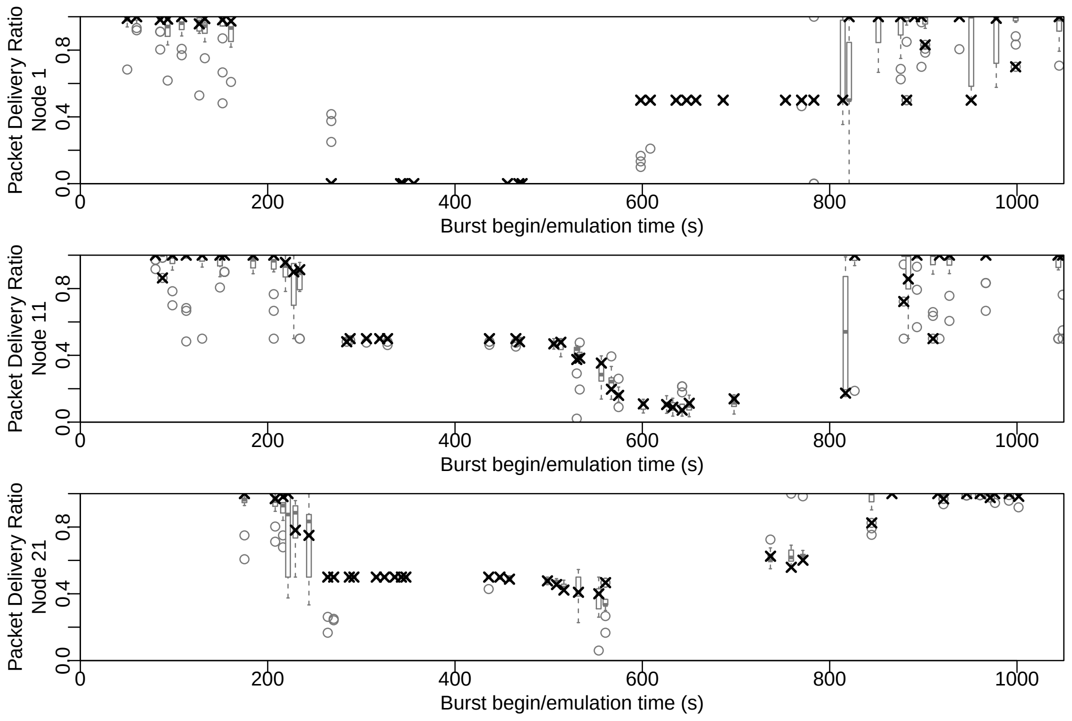




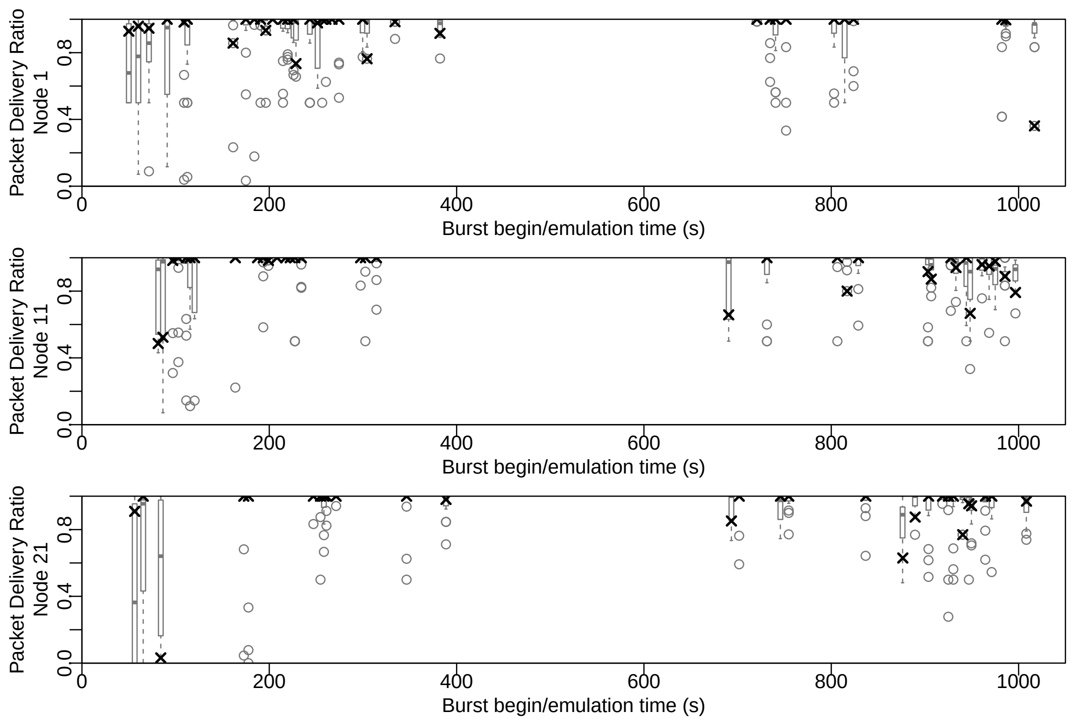
Scenario 16



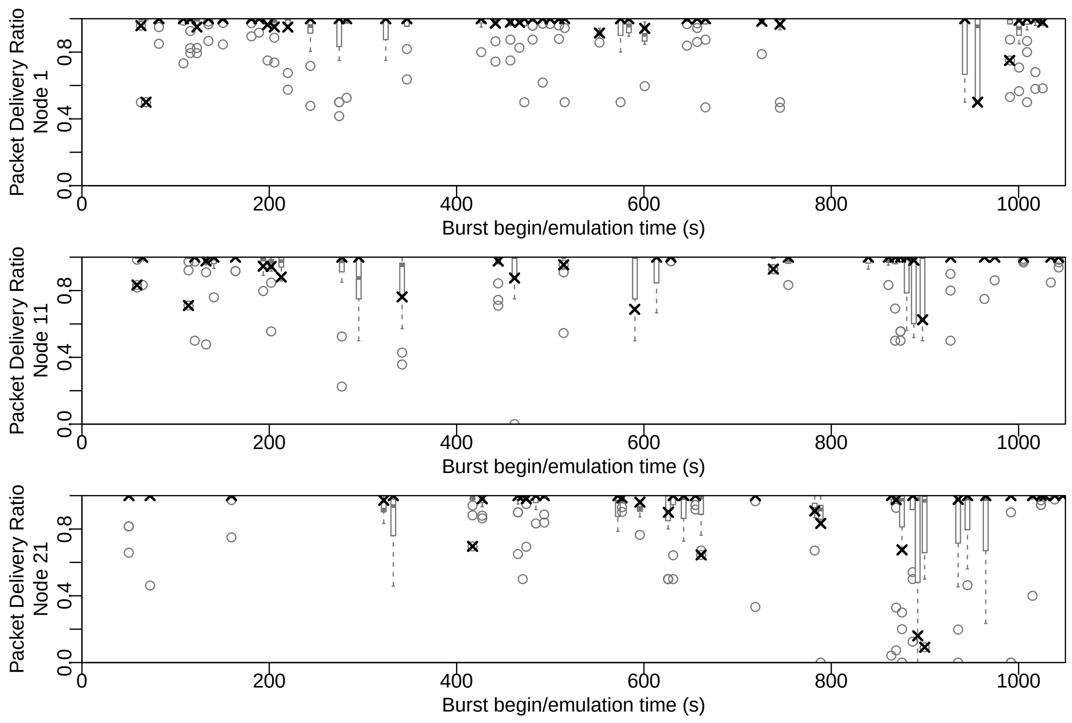
Scenario 17



Scenario 18

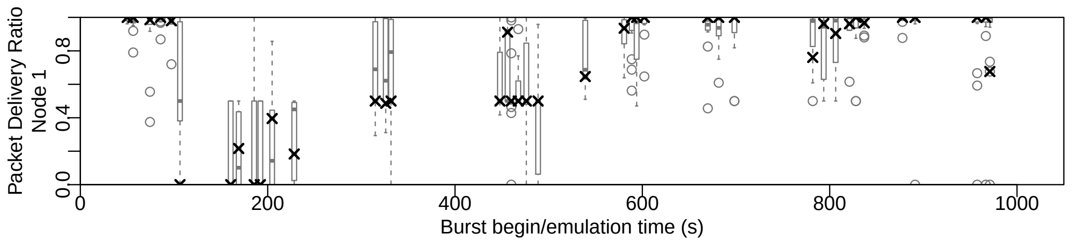


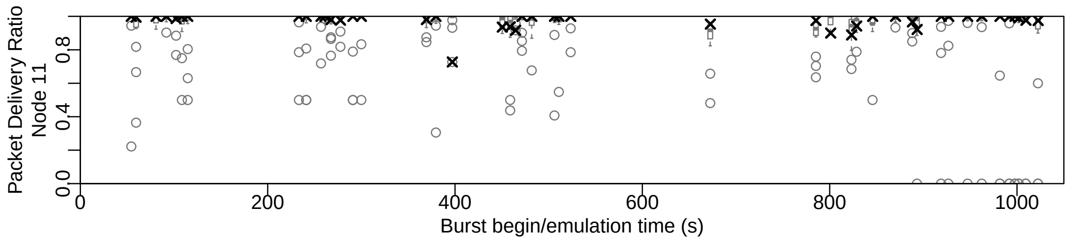
Scenario 19

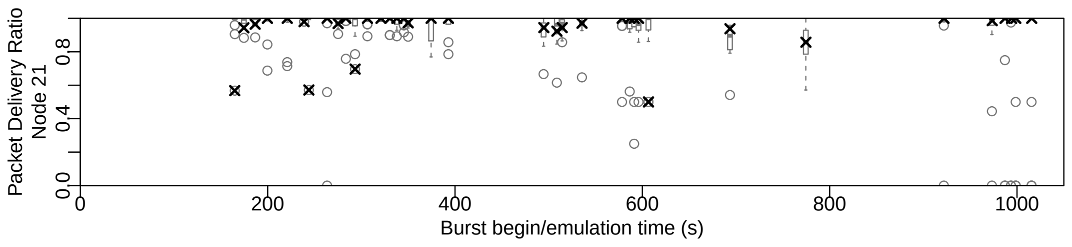


B.2. Packet Delivery Ratios in Emulations With Attacks

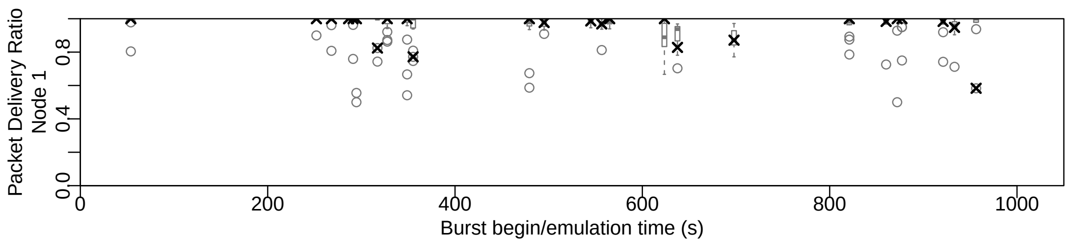
Scenario 0

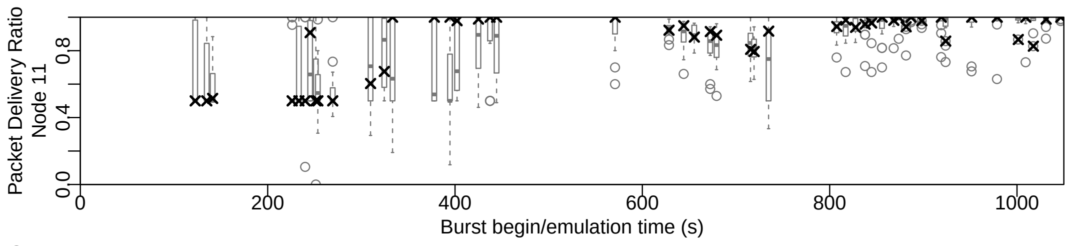


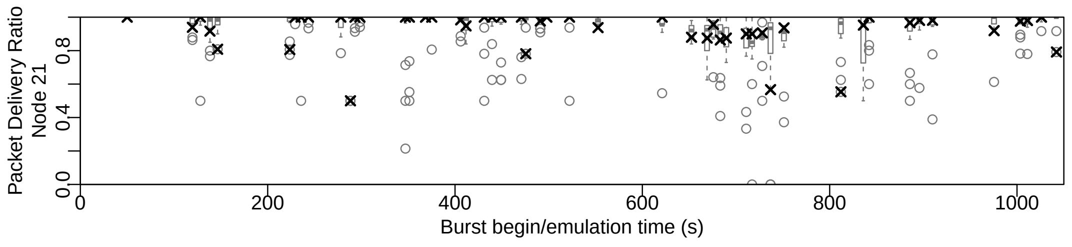




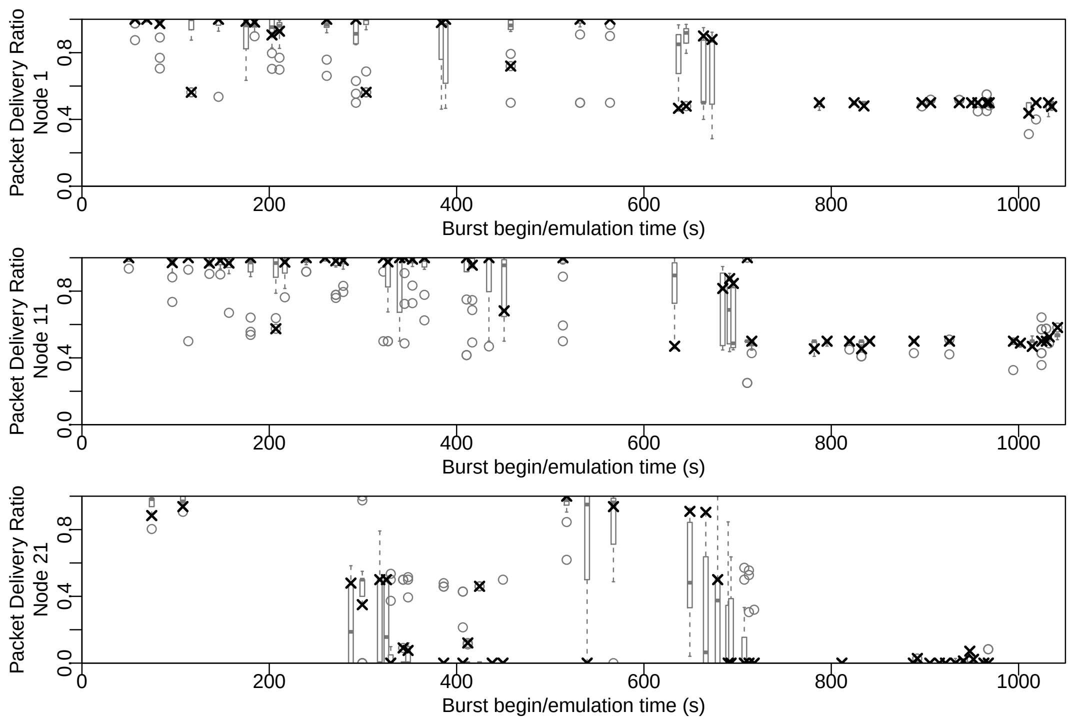
Scenario 1



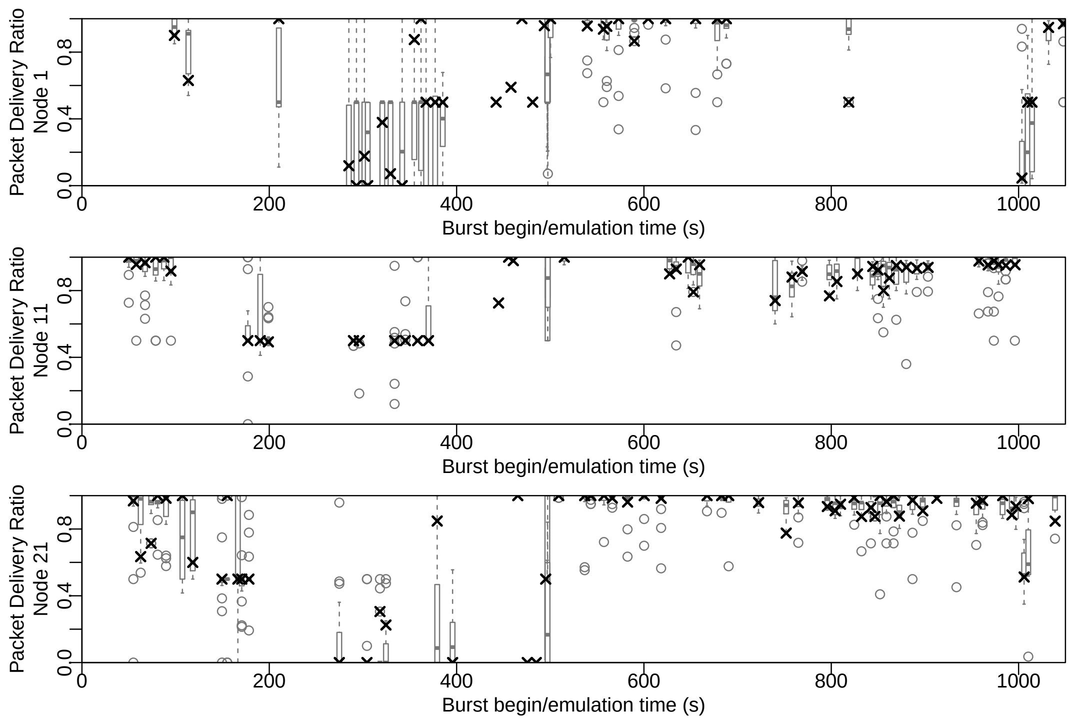




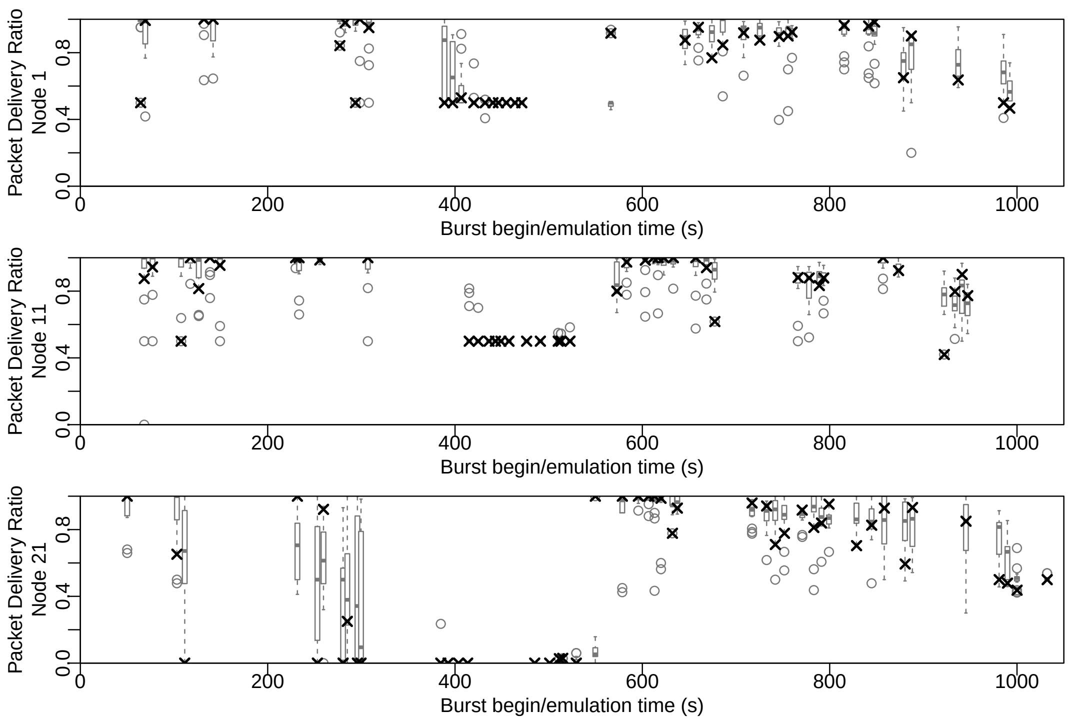
Scenario 2



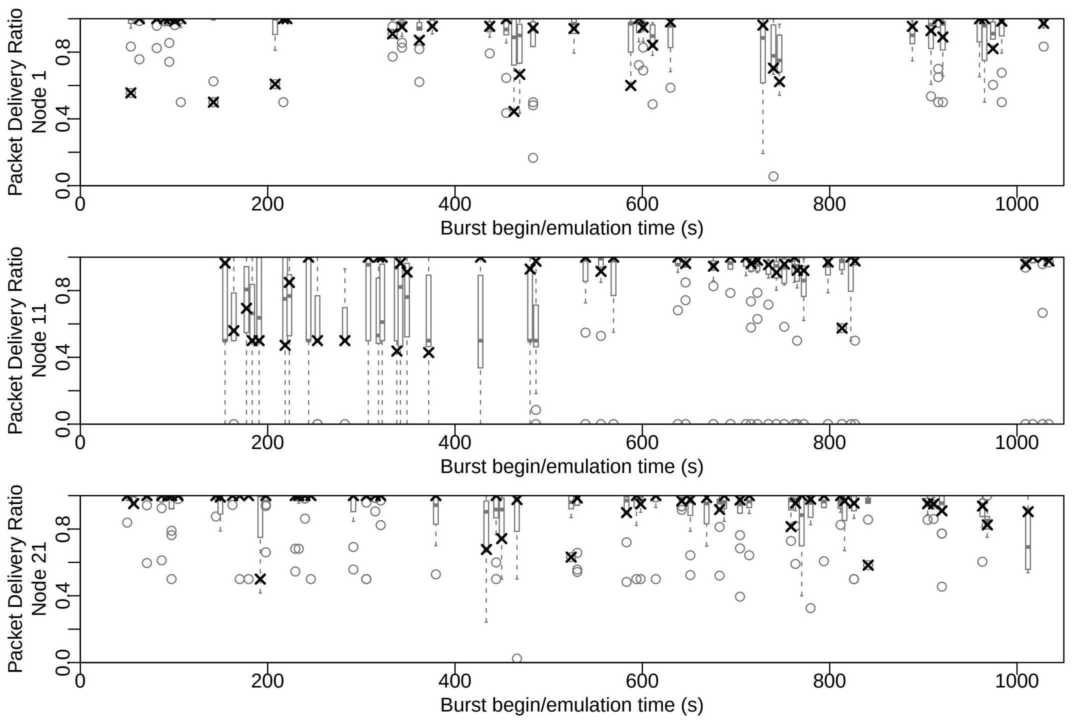
Scenario 3



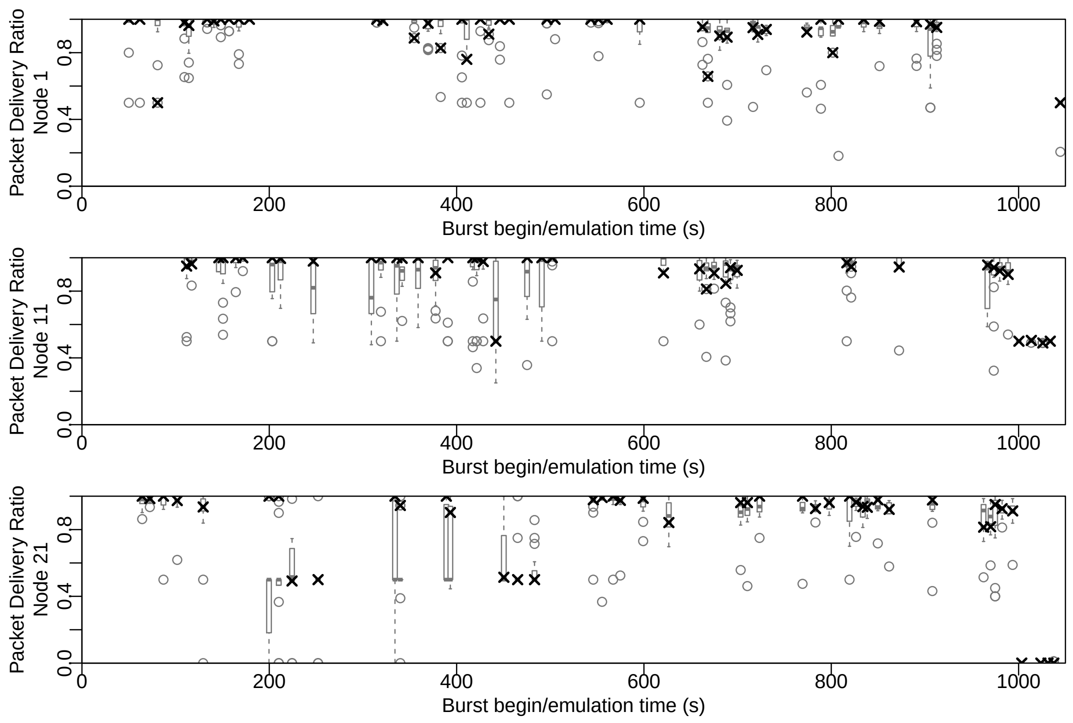
Scenario 4



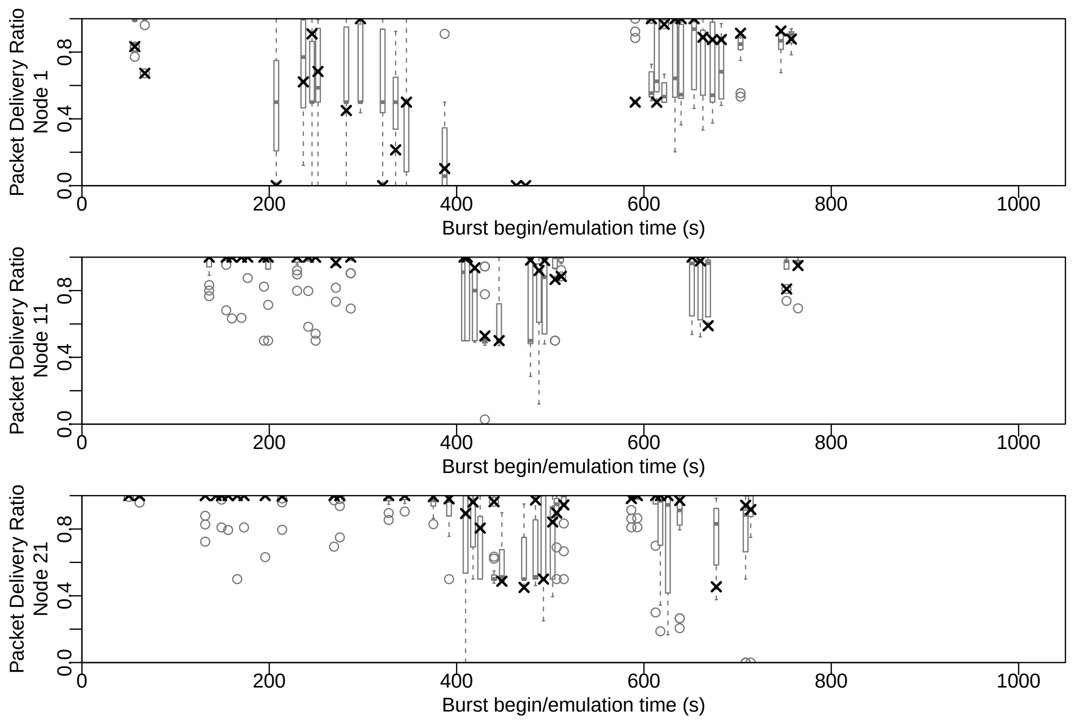
Scenario 5



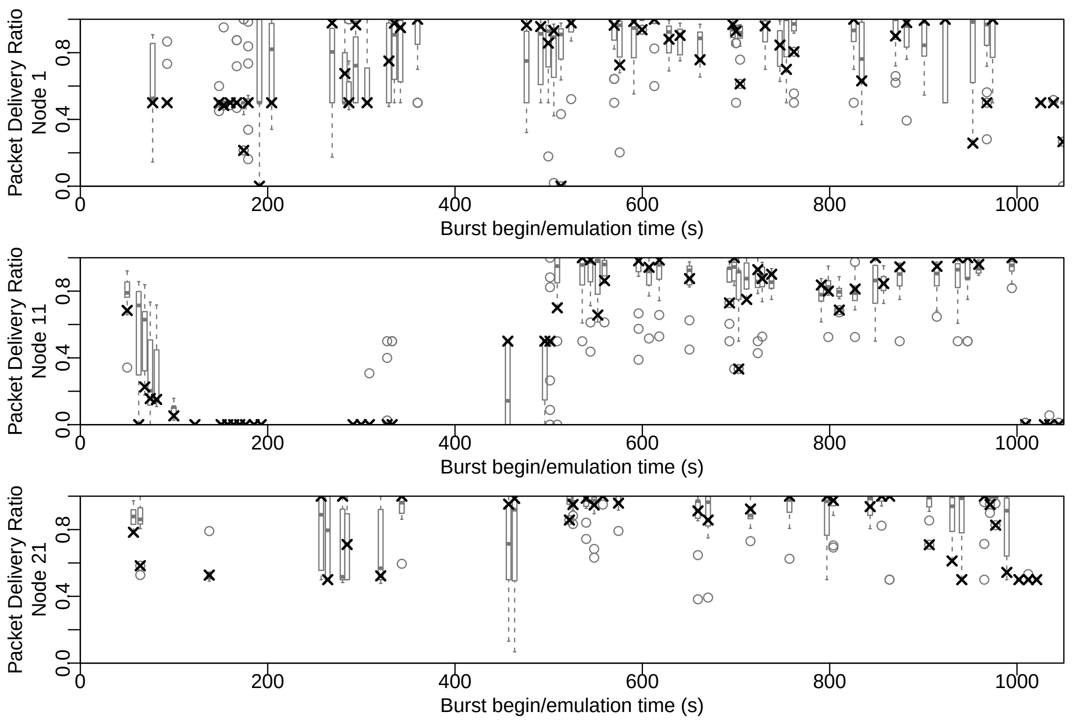
Scenario 7



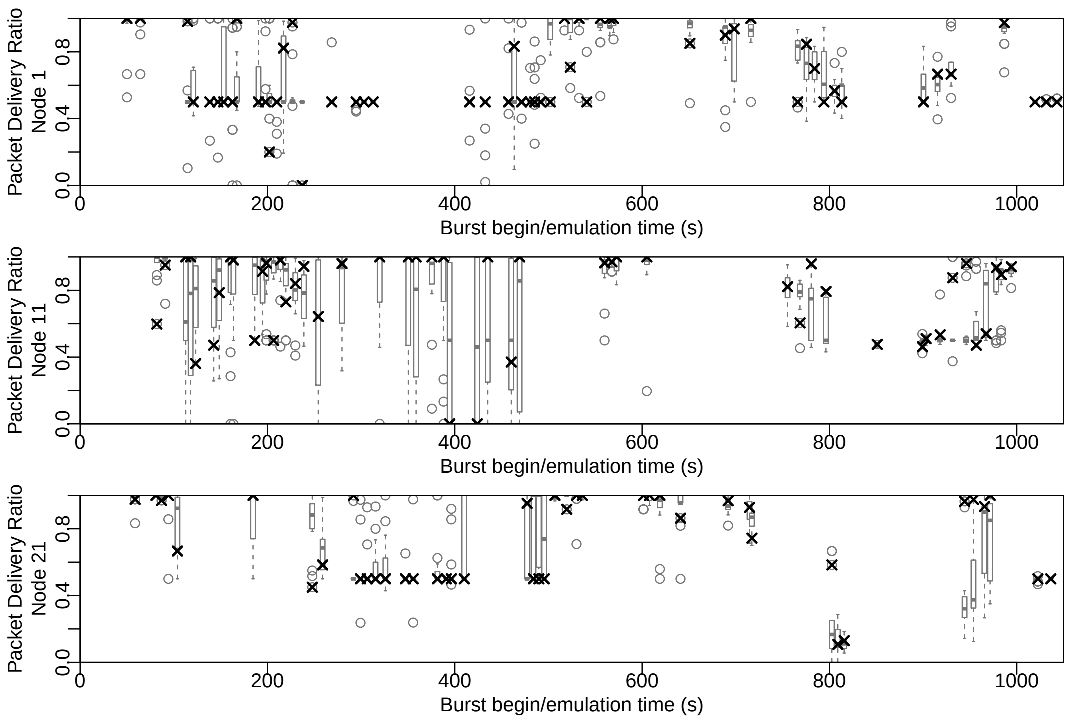
Scenario 8



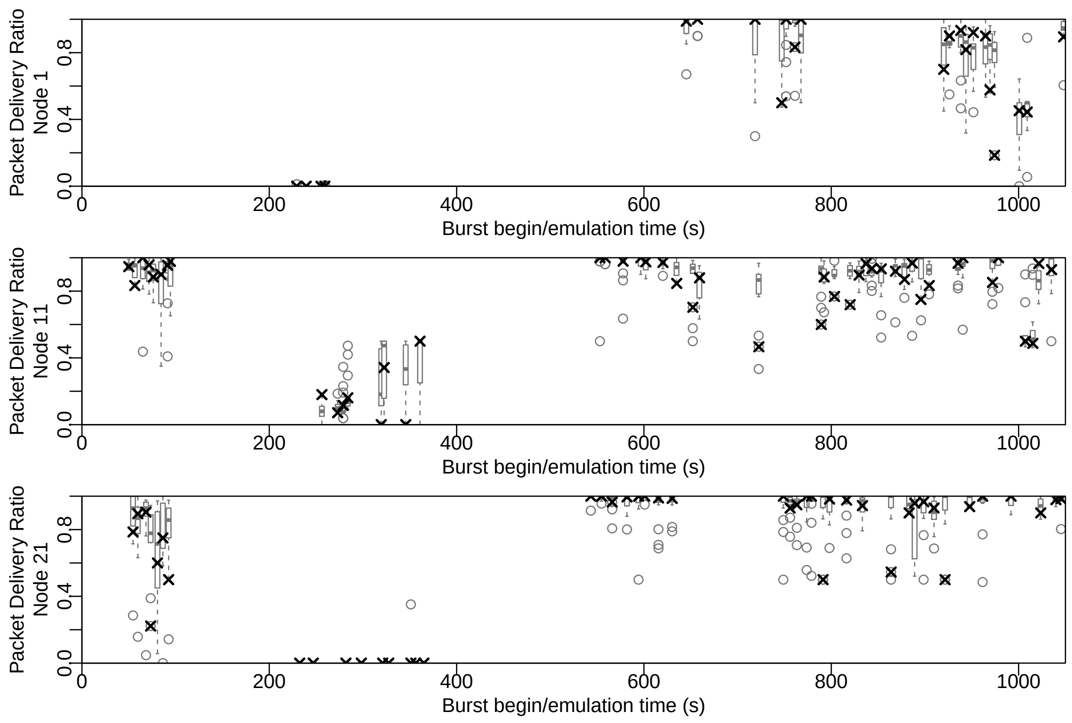
Scenario 9



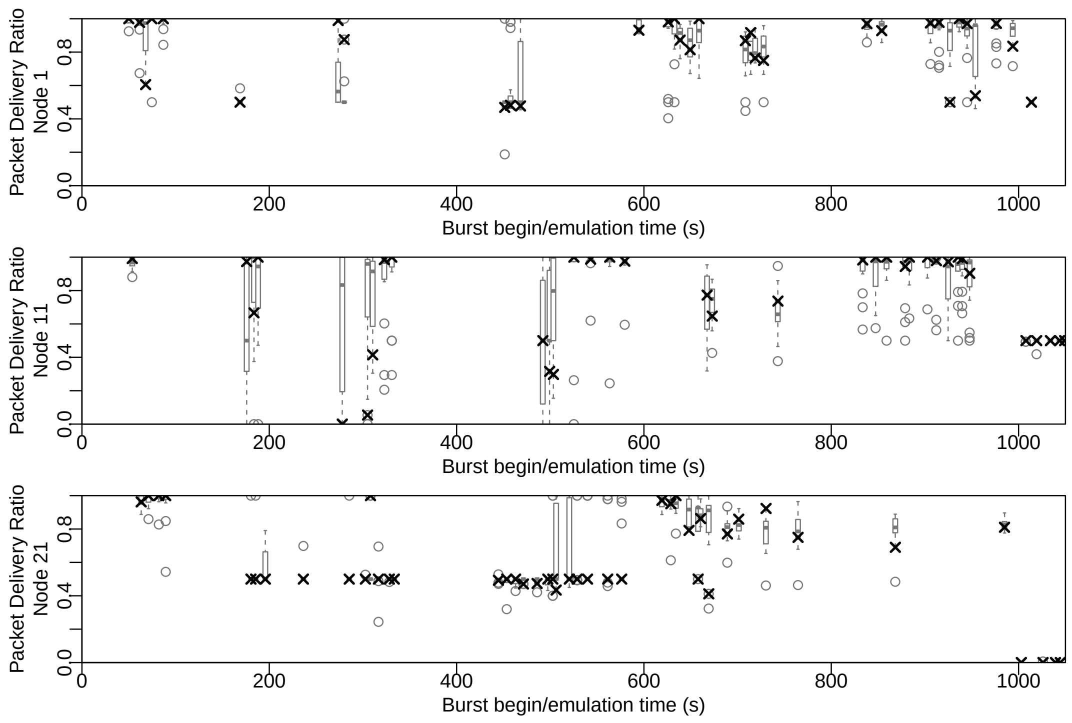
Scenario 10



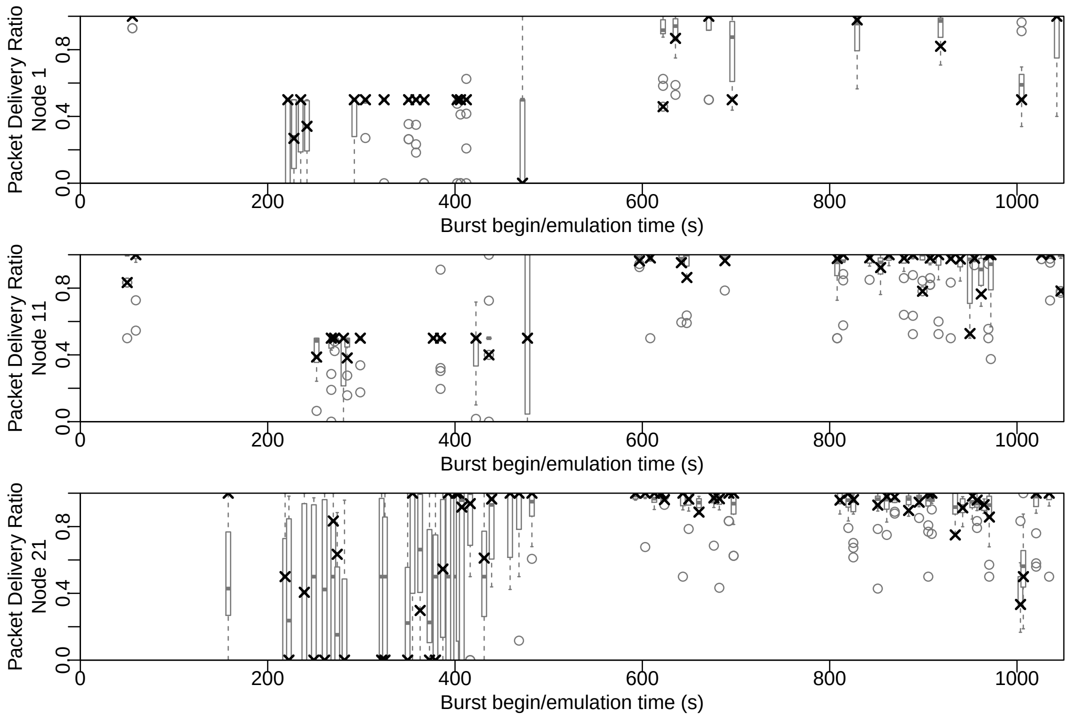
Scenario 11



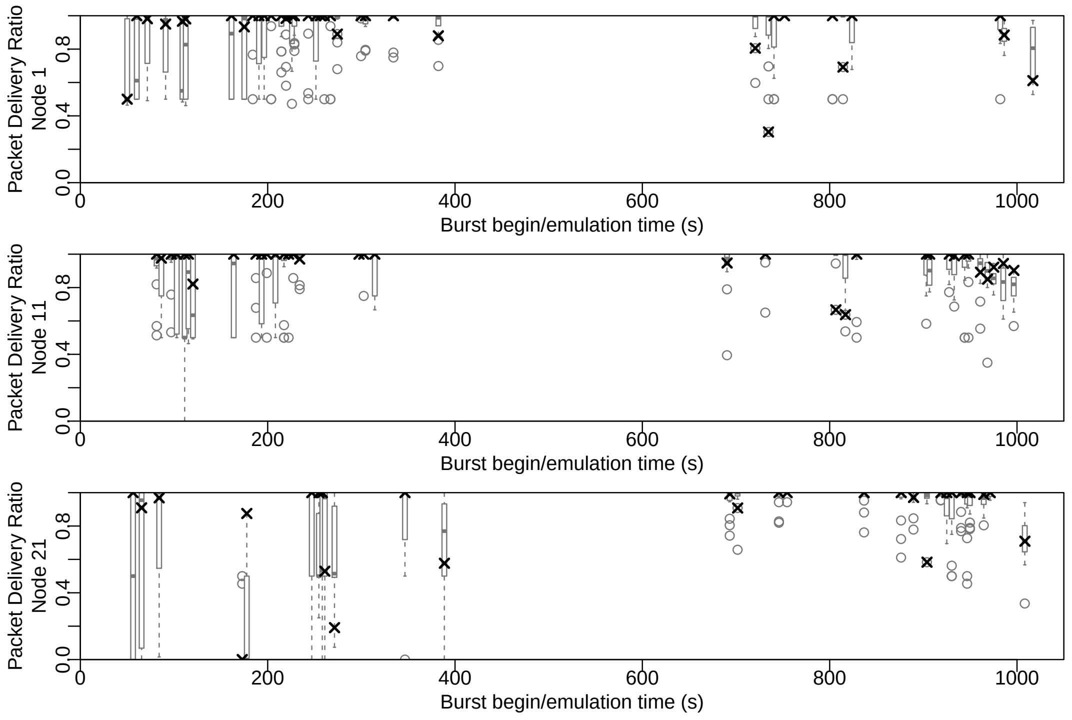
Scenario 14



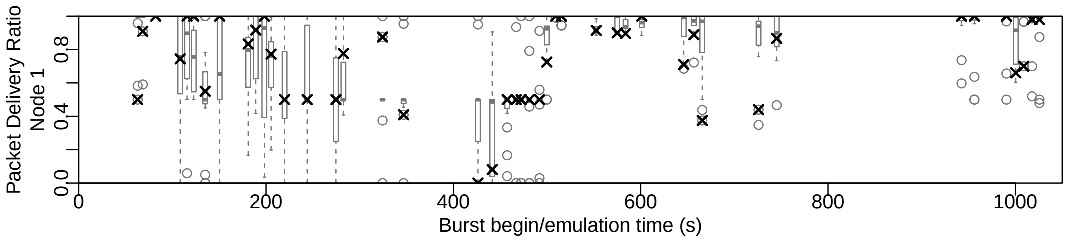
Scenario 15

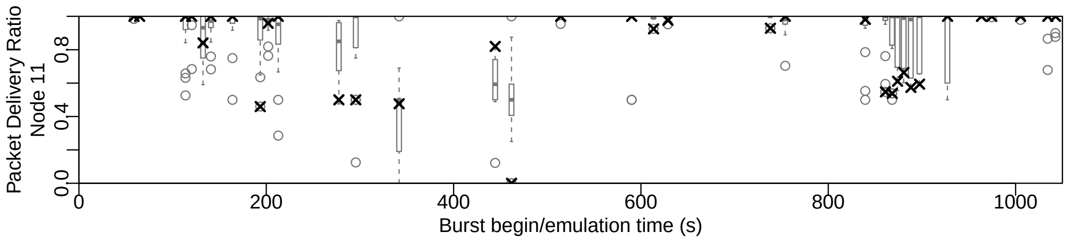


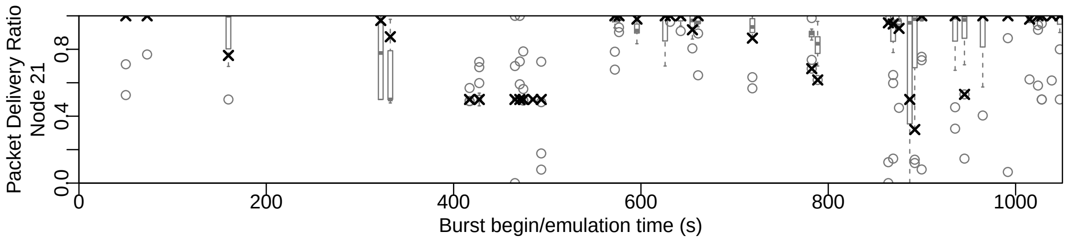
Scenario 18



Scenario 19


Packet Delivery Ratio
Node 1
0.0
0.4
0.8
0
200
400
600
800
1000
Burst begin/emulation time (s)


Packet Delivery Ratio
Node 11
0.0
0.4
0.8
0
200
400
600
800
1000
Burst begin/emulation time (s)

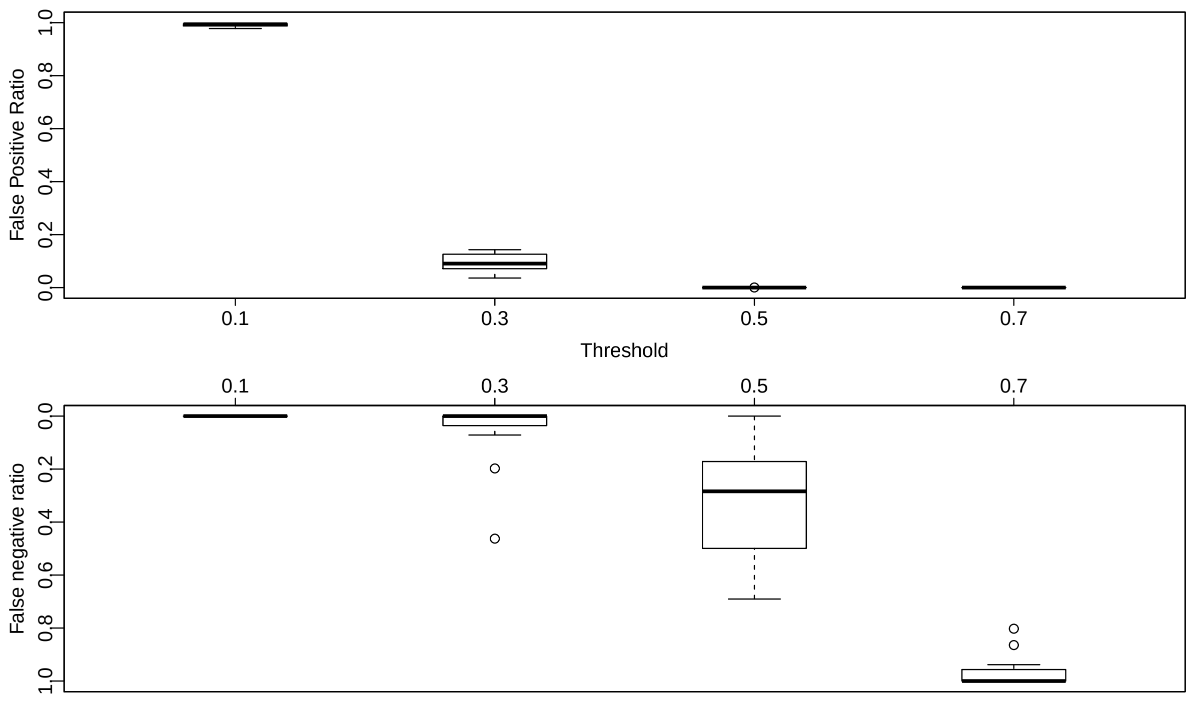

Packet Delivery Ratio
Node 21
0.0
0.4
0.8
0
200
400
600
800
1000
Burst begin/emulation time (s)

Appendix C

Additional Evaluation Results

C.1. ETXBAD

Mean with Node Threshold



This graph presents the false positive and negative ratios for the ETXBAD detector with minReporters set to 5. Since the numbers obtained in are identical to the ones discussed in Section 7.3.3, we abstain from discussing the results here.

Appendix D

Challenge-LQ Attacker Plug-in Control Message Format

Our attacker control message is contained in the payload of a regular OLSR message with the type field set to 155. It will only be parsed by plug-in instances that have been compiled with attacker extensions being enabled. The message payload contains a sub-header including a type, reserved and a multiplexer field, as indicated by table D.1. By setting the type field, the user indicates whether the message should change the configuration for forging neighbourships,

Offset	Byte 0	Byte 1	Byte 2	Byte 3
+ 0 bytes	Type	Reserved	Multiplexer ID	

Table D.1.: The Challenge-Response plug-in's attacker control message subheader.

Offset	Byte 0	Byte 1	Byte 2	Byte 3
+ 0 bytes	Flags	Link Quality	Neighbour Count	
+ 4 bytes	Start Time, Seconds Part			
+ 8 bytes	Start Time, Microseconds Part			
+ 12 bytes	Stop Time, Seconds Part			
+ 16 bytes	Stop Time, Microseconds Part			
+ 20 bytes	First Address in Range...			
+ 24 bytes	...First Address in Range...			
+ 28 bytes	...First Address in Range...			
+ 32 bytes	...First Address in Range			
+ 36 bytes	Last Address in Range...			
+ 40 bytes	...Last Address in Range...			
+ 44 bytes	...Last Address in Range...			
+ 48 bytes	...Last Address in Range			

Table D.2.: The message format of a neighbourship forgery submessage.

Offset	Byte 0	Byte 1	Byte 2	Byte 3
+ 0 bytes	HELLO-Flag	Link Quality	Reserved	
+ 4 bytes	Start Time, Seconds Part			
+ 8 bytes	Start Time, Microseconds Part			
+ 12 bytes	Stop Time, Seconds Part			
+ 16 bytes	Stop Time, Microseconds Part			

Table D.3.: The message format of a link quality forgery submessage.

i.e. one of the first two modes, or for forging existing neighbourships' link qualities. While the reserved field merely serves the purpose of padding the header to multiples of 32 bits, the multiplexer allows choosing the addressee of a given message.

The requirement for the multiplexing feature stems from the delivery method of the attacker control messages. To use the messaging provided by the OLSRd, we implemented, as indicated above, the attacker control messages as an OLSR message with a custom type ID. Since the OLSRd is however supposed to only consider traffic concerning the routing, it will only listen on the local broadcast address of a given device. Particularly, it will not receive any packets that were addressed to a specific IP address. To send an attacker control message to an instance of the OLSRd, it has to be sent to the respective broadcast address, even when the respective packet would originate from the very same node. An instance of the attacker plug-in would thus not be able to determine whether it was supposed to process a control message it received or not. To ensure that we would not be limited to running a single attacker at a time without risking configuration issues, we hence included the multiplexing feature. When a user sets up a multiplexer ID through the OLSRd's configuration file, upon receiving a control message, the plug-in will compare the stored ID versus the one advertised in the message and silently discard it, if they differ, if no multiplexer ID was configured, it defaults to 0.

As indicated above, both methods for forging neighbourships described in Section 6.1.2 use the same message format. To indicate that the following message is of that type, the control message header's type field has to be set to 2. Table D.2 indicates the format of the respective payload. It starts with a flags field which can be used to toggle the following features:

- Forge HELLOs only
- Spoof TC messages from forged neighbours
- Set the TTL of spoofed TC messages to 1

While the purpose of the first flag may be obvious, an attacker may want to limit the scope of his attack to his immediate neighbours to complicate its detection, the latter are not. During the course of this thesis, we discovered that the OLSRd would only consider such links in its routing process that had been advertised in TC messages and by both adjacent nodes, unless the link in question was between itself and the given node. This would render a sinkhole attack ineffective under all but some very specific conditions. Since this was not intended by the OLSR Request for Comments (RFC), we decided to sidestep this implementation issue by adding a feature to our plug-in that would send spoofed TC messages, claiming to originate from the forged neighbours and ensuring that the requirement of the routing process would be met. As for HELLO messages, an attacker might want to limit the scope of the respective messages to its own neighbours to render detection more difficult, hence the message contains, along with the flag for enabling the spoofing feature, another flag allowing to indicate whether the spoofed TC messages should contain a TTL of 1, which would result in the attacker's neighbours discarding the message after processing it.

The second field in the neighbourship forgery message indicates the link quality value that should be advertised for forged neighbours. After that follows a field indicating the count of neighbours to forge. If this field's most significant bit is set to 1, the attacker plug-in will forge in accordance with the first of the two forgery modes indicated above, i.e. claim neighbourship to all addresses in a range provided. With any other value, it will follow the second paradigm and thus forge the given count of neighbours, choosing addresses from the given range but skipping addresses of nodes that are already neighbours. Note that in the latter mode, the attacker may only be able to forge less than the given number of neighbourships, if the given address range, after removing the existing neighbourships, holds too few addresses.

After that come four 32 bit unsigned integer fields encoding two microsecond precision timestamps. The attack will only be perpetrated for messages which are sent in between the first and second timestamps, i.e. these fields can be used to send the respective OLSR message early and ensure that an exact time frame is observed for executing the attack. Finally, two fields of 128 bits each hold the first and last address of the range that forged addresses should be forged. If the OLSRd is running in IPv4 mode, only the first 32 bits of each field will be considered, interpreting them as IPv4 addresses, otherwise, the whole field is considered for each address. Constructing addresses from a range simply interprets the value of “First Address” as an unsigned integer of the appropriate length and increments it in steps of one until either the requested number of addresses has been generated or the value of “Last Address” has been returned as an address.

With submessage type 1, the user may set up forging of link qualities for existing neighbourships. The respective message follows the same principles, but requires less options and thus uses the simpler message type reflected in table D.3. It starts with a flag field which only distinguishes between forging link qualities in all messages (when set to 0) or restrict it to HELLO messages (any other value). The next field indicates the link quality value that should be advertised for the respective nodes and is followed by a 16 bit reserved field to padd the message to multiples of 32 bits. Finally, this submessage also comes with a microsecond precision start and end timestamp encoded in two 32 bit unsigned integers each. As for the neighbourship forgery message, it allows configuring the time frame during which the attack should be perpetrated.