BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS
ÚSTAV TELEKOMUNIKACÍ

AD HOC NETWORK SCENARIOS AND DIFFERENT TYPES OF ATTACKS
USPOŘÁDÁNÍ AD HOC SÍTÍ A RŮZNÉ TYPY ÚTOKŮ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE
Denis Rexa

SUPERVISOR
VEDOUCÍ PRÁCE
doc. Ing. Vladislav Škorpil, CSc.

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Denis Rexa ***ID:*** 164383
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Uspořádání ad hoc sítí a různé typy útoků

POKYNY PRO VYPRACOVÁNÍ:

Věnujte se vlastnostem MANET sítí a možnostem simulačního programu NS3. Prostudujte a teoreticky zpracujte nejrozšířenější hrozby pro MANET (ad hoc) sítě a popište problém "broadcast storm" v ad hoc sítích. V další části práce MANET síť simulujte, demonstrujte útoky na datový provoz a v dané topologii je ovlivňujte. Na základě získaných poznatků navrhněte možnosti snížení dopadu útoků na síť.

DOPORUČENÁ LITERATURA:

[1] BHATTACHARYYA, A., BANERJEE, A., BOSE, D., SAHA, H., N., BHATTACHARJEE, D. Different types of attacks in Mobile ADHOC Network: Prevention and mitigation techniques. https://ipfs.io/ipfs/QmfXH9XtP7xmoTH8WAp4HNSduqWMwLTH8B8TvbTkdgzNAa/cc-by-nc-sa-3.0/1111/1111.4090.pdf

[2] LI, W., JOSHI, A. Security Issues in Mobile Ad Hoc Networks - A Survey. http://www.csee.umbc.edu/~wenjia1/699_report.pdf

[3] ŞEN, S., CLARK, J., A., TAPIADOR, J., E... Security Threats in Mobile Ad Hoc Networks.

https://www-users.cs.york.ac.uk/~jac/PublishedPapers/SecurityThreats%20in%20MANEs.pdf

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 6.8.2017

Vedoucí práce: doc. Ing. Vladislav Škorpil, CSc.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

The research described in this bachelor thesis has been done in laboratories supported by Sensor, Information and Communication Systems Research Centre (SIX) project; registration number CZ.1.05/2.1.00/03.0072 Operational Program Research and Development for Innovation (operační program Výzkum a vývoj pro inovace).

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

Abstrakt

V tejto práci venovanej sieťam MANET a ich bezpečnostným hrozbám sa v prvej kapitole preberajú vlastnosti sieti MANET a porovnávajú sa smerovacie protokoly AODV vs OLSR vs ZRP ako zástupci proaktívneho, reaktívneho a hybridného smerovania so spomenutou hlavnou charakteristikou s výhodami a nevýhodami jeho využitia.

Ďalej je spomínaný "broadcast storm" a podrobnejšie rozobrané základné typy útokov v sieťach MANET rozdelené do skupín Data traffic (Black-hole, Cooperative Black-hole, Gray-hole) a Control traffic (Hello flood, Wormhole a Rushing Attack) s návrhmi na potlačenie vplyvu na fungovanie siete.

V nasledujúcej kapitole je ukazané a popisané prostredie NS-3 simulátora, konkrétnejšie vo verzii 3.21 s uvedeným jednoduchým programom, z ktorého som neskôr vychádzal + spôsoby, ako získať údaje zo simulácie + ukážka topológie mnou vytvorenej siete v nadstavbe NetAnim vo verzii 3.105.

Posledná kapitola ukazuje dopad black-hole a gray-hole útoku na sieť s vlastným návrhom implementácie v zdrovojom kóde. V záveru každej podkapitoly je spomenutý návrh na potlačenie vplyvu útoku na sieť.

Cieľom tejto práce je vytvorenie simulovanej siete s implementovaným ad-hoc smerovacím protokolom s viacerými uzlami s nastavenou fixnou mobilitou (keďže pozorovanie pohybových modelov nie je cieľom tejto práce) a skúmať dopad black-hole and gray-hole útokov na sieť. Pri simulacií black-hole (BH) som využil ako základ jednoduchú sieť s dodaním možnosti aktivácie BH pomocou príkazového riadku. V tom prípade napadnutý uzol v nastavený čas začne nepresmerovávať pakety, sieť je teoreticky rozdelená na dve časti a strata paketov pri pokuse o preposlanie cez napadnutý uzol je rovná 100%. Pri aplikovaní BH som nešiel cestou stopnutia presmerovovávania paketov, ale „zhodenia" uzlu. Keďže som sa po mnohým bezúspešných pokusoch o implementáciu správnej funkcie na fórach ns-3 užívateľov dozvedel, že daná funckia SetForwarding nefunguje správne a jedná sa chybu softwaru.

Pri simulácií gray-hole útoku (GH) som sa sústredil hlavne na môj vlastný návrh implementácie do zdrojového kódu. Vybral som si variantu – „node dependency" attack kedy sa pakety filtrujú na základe IP adresy alebo aj portu. Keďže som neobjavil funkčný spôsob akým možno filtrovať komunikáciu medzi uzlami generovanú aplikáciou, bol som sa nútený pristupovať k paketom ako objektom a generovať ich manuálne. Musím priznať, že tento spôsob plne neodpovedá myšlienke GH, keďže som posielal pakety cez uzol, ktorý pakety filtroval, ale vytvoril som ho ako server. To znamená, že pri preposlaní do cieľovej adresy v hlavičke paketu figuruje cieľová adresa prostredného servera a nie skutočného odosieľatela. Výsledky práce nemám vyjadrené percentuálne, keďže pakety boli generované manuálne a tým pádom nešlo o plnohodnotnú komunikáciu medzi účastníkmi.

Kľúčové slová

MANET, AODV, OLSR, ZRP, NS-3, NetAnim, Black-Hole, Gray-Hole

Abstract

In this thesis dedicated to MANET networks and security threats are compared the routing protocols AODV vs OLSR vs ZRP. Next chapter is dealing with a broadcast storm and discussed in more detail basic types of attacks in MANET networks with proposals to suppress the impact on the functioning of the network. In the next chapter, it is shown and described an environment of NS-3 simulator with that program and simple way how to reach the information. The last chapter shows the influence of black-hole and gray-hole attack on the network, and a proposal for its elimination.

Key words

MANET, AODV, OLSR, ZRP, NS-3, NetAnim, Black-Hole, Gray-Hole

Bibliografická citace mé práce:

REXA, D. *Ad Hoc Network Scenarios and Different Types of Attacks.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 48 s. Vedoucí bakalářské práce doc. Ing. Vladislav Škorpil, CSc..

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Ad Hoc Network Scenarios and Different Types of Attacks jsem vypracoval samostatně pod vedením vedoucího semestrální práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené semestrální práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne…….

…….

podpis autora

Content

Introduction

Mobile ad-hoc network (MANET) is getting increasingly used on daily based. People require to be independent on fixed infrastructure and be online whenever they want to be. In my thesis will be discussed advantages and disadvantages of this kind of network and threats connected with it. As a main characteristic feature, I would mention that MANET is self-configuring system without any base station.

The goal of this thesis conveys sample of the impact of attack (respectively the threat in general), on the topology in the simulator NS-3 (free software, available under the GNU GPLv2 license for research, development and use). It doesn't dispose a graphical user interface in which the simulation could be created, but it's running in the form of source code, which is based on C ++ or Python.

First chapter is dealing with MANET network basic features, routing techniques using within the whole internet. There are compared routing protocols designed for MANET, which are divided among 3 groups according to their behavior.

In Next chapter, there was chosen a representative from every group of MANET protocols and this chapter devoted to comparison AODV vs OLSR vs ZRP summarizes main characteristics of each protocol. There are chosen also recommendation contained for which types of network protocols will be given protocol the most effective.

The following chapter theoretically discusses the impact of broadcast storm on the topology and are mentioned also attacks such as black-hole or gray-hole with proposals to eliminate their influence.

The most extensive chapter is devoted to creating a model in the network simulator, implementation of applications with setting of parameters and there is also the description of creation individual parts of the network. Everything is detailed shown and explained on the code sample.

The last chapter is dealing with the impact of black-hole and gray-hole attack on continuity of packets flowing and it contains a proposal how to mitigate the influence of the attack.

1 MANET network

Perhaps the most widespread notion of a mobile ad hoc network is a network formed without any central administration which consists of mobile nodes that use a wireless interface to send packet data. First mention of ad hoc networking can be found in late 60's, when work on the ALOHA network was established (the objective of this network was to connect educational facilities in Hawaii). [3] The ALOHA protocol worked as a single hop protocol without any routing option, not suitable for commercial using. In 1973, DARPA initiated work on the PRnet (packet radio network)—a multihop network [11]. In the 80's, DARPA declassified it for no military purpose.

We understand by the term ad-hoc that two equal nodes establish the connection temporarily without any centralized node or base station. Self-configuring feature is one of the most significant criterion because the environment and topology is changing constantly and dynamically. It is well known and widely applied in military purpose, disaster area, PAN networks etc. [9] First ad-hoc network was build up in 70's, however, the mobility of system was not integrated into the system.

1.1 MANET features

This network is such as flexible solution, therefore it's suitable for areas where developing the infrastructure is not worth. A mobile ad hoc network (or simply MANET throughout this paper) has some special features that we should take into account while developing the algorithm:

- It is a self-configuring network which is composed of several movable user equipment.
- It does not rely on any fixed infrastructure, connected by wireless links/cables
- lack of base station or centralized administration
- unidirectional or bidirectional connection
- disadvantages of wireless networks- sensitive to noise and interference
- fluctuating link capacity: the effects of high bit-error rates might be more noticeable in a multihop ad hoc network
- MANET node must rely on itself as regards a security or routing
- low-power devices: CPU processing, memory size/usage, signal processing and transceiver output/input power. These are power-consuming components and applied algorithms or mechanisms should be optimized to mitigate power consumption [11]

1.2 Routing in general

Routing is described as a process of selection best path how to get to the destination. What is the most suitable path the router decides with implemented routing protocols. The routing process describes the way of forwarding packets through the network based on routing, which maintain a record of the routes to different network destinations. Constructing routing tables held in the router's memory and their constant updating is very important for efficient routing.

1.2.1 Static routing

Routing method when router uses a manually-configured routing entry. Routers do not examine the network, they don't communicate with each other and their routing table are built up by administrator by adding in entries into. Unlike dynamic routing, the routes are fixed and if topology changed, the administrator will have to rebuild the routing table every time. This technique is employed very rarely, because of a duration of rebuilding.

1.2.2 Dynamic routing

Unlike static routing, dynamic routing enables routers to select paths according to real-time topology changes. Router is responsible for the creation, maintenance and updating of the dynamic routing table and resend the actualization to its neighbours.

We can divide them into two essential group:

- IGP (Interior Gateway Protocol)- for routing within the autonomy area; protocols RIP, EIGRP, OSPF
- EGP (Exterior Gateway Protocol)- for routing out of the autonomy area; protocols BGP, EGP3 (it won't be mentioned below)

Routing protocols IGP

Distance Vector (DV):

- best way= lowest number of hops
- sending the routing table periodically
- it doesn't know whole topology
- lower performance needed

Link State (LS):

- it knows whole topology
- maintain topology database, which are updated via LSA (Link-State Advertisement) messages
- higher performance for processing the received information needed
- service messages for updating information are sent only if some topology changes or failure occur

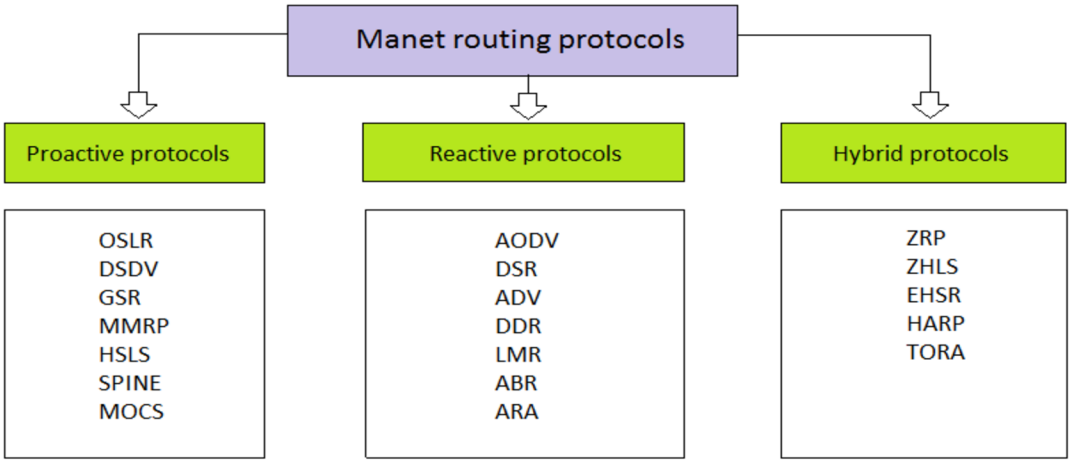
1.3 Routing in MANET

In extensive topologies might happen that between two different nodes can exist more than one path, known as redundancy. For cases if some troubles with the nodes occur, packet will be sent via backup path. Routing within MANET network faces to challenging task because the nodes can move randomly within the network. An optimal path to destination are changing every moment, and given path consider as a best one might not work several seconds later.

Likely dynamic protocols, we can divide them into two essential group:

- Connection-oriented protocols- the path is known before sending data
- Connectionless protocols – the optimal path will be chosen after sending and the source node don't need to obtain information about successfully received data

MANET routing protocols are divided among: 1) table-driven (proactive) routing, 2) on-demand (reactive) routing, 3) hybrid (both proactive and reactive) routing



Pict. 1: Dividing the protocols

1.3.1 Proactive routing

Proactive, in terms of routing in MANET, means each node is actively mapping the topology and the paths to the destination are selected before sending data- compiled routing tables which are periodically updated. The individual routes are calculated, even if that do not affect the functioning of the network. [6]

Node contains the number of hops required to arrive that particular destination node, generation of new sequence number marked by the destination and the destination address. The proactive protocols are appropriate for less number of nodes in networks what result in more maintenance traffic for updating table of every node. It might cause Routing overhead problems. [13] Among proactive protocols belong protocols such as DSDV, OLSR, FSR and WRP.

1.3.2 Reactive routing

Unlike active routing, in reactive protocols, routes are determined on demand. It uses flooding concept and exploring the topology in order to get latest routing table is not required. Finding out process for the optimal path to the destination works on the principle the flooding request message (RREQ) and reply (RREP). No need to look the way for each packet, so that it is assumed that the path is available throughout the broadcast. [6]

We have to take into account higher delay because of on demand routing. But on the other hand, reactive protocol has lower overhead with minimum service messages-> not wasting with bandwidth. Among reactive protocols belong protocols such as DRS, AODV and ADV.

1.3.3 Hybrid routing

This combines the advantages of both- proactive and reactive protocol. Created for purpose to reduce the control overhead traffic of proactive routing protocols and also decrease the latency caused by route discovery in reactive routing protocols. [14] The nearby nodes are trying to find ways immediately, which are regularly updated. Routes to remote nodes are created at the moment when it is required [6]. Hybrid protocols are for instance ZRP (Zone routing protocol) and TORA (Temporarily Ordered Routing Algorithm).

2 AODV vs OSLR vs ZRP protocols

It was chosen a representative from every group and their features are going to be compared in following chapter.

2.1 AODV protocol (Ad-Hoc On Demand Distance Vector Routing)

AODV (Ad hoc On-Demand Distance Vector) was developed on July 2003 in Nokia Research Center, University of California, Santa Barbara and University of Cincinnati by C. Perkins, E. Belding-Royer and S. Das. [15] As it was mentioned, routes are determined on demand, which causes control traffic overhead to be dynamic. It will result in initial the delay.

2.1.1 Main characteristic of AODV

- Once route is established between two endpoints, AODV remains passive. In the moment the route becomes invalid or lost, AODV will demand a request.
- Exploring the best path is initiative by broadcasting HELLO message with set TTL (Time To Live) value to 1 what ensures receiving only by the neighbours. These messages are going to be forwarded to each of remote neighbours (imagine it as a tree structure), until one of them does not response with positive answer- it knows the way. RREQ packet is marked where it came from. On the base of this data is created “map” and sent the answer- RREP (Route Reply).
- RERR messages (Route Error) serves to notice neighbours the path is going to become invalid or time will expire soon. Messages took part on the forwarding to clear all roads that were using the questionable part of the route [6]

2.1.2 Advantages of AODV

- Less system messages
- More suitable for smaller topologies
- sequential numbers guarantee that the information is up to date [6]

2.1.3 Disadvantages of AODV

- Delay of packets
- Demanding implementation of QoS – RREP and RREQ have to carry more information (bandwidth, delay,…) [6]

2.2 OLSR (Optimized Link State Routing)

OLSR is a proactive link-state routing protocol, which uses hello and topology control (TC) messages to discover and then disseminate link state information throughout the mobile ad hoc network. Individual nodes use this topology information to compute next hop destinations for all nodes in the network using shortest hop forwarding paths. [17]

Using Hello messages, the OLSR protocol at each node discovers 2-hop neighbour information and performs a distributed election of a set of multipoint relays (MPRs). [17] MPRs node must comply with conditions such as existing a path to each of its 2-hop neighbours via this node. Their role is to source and forward TC messages.

2.2.1 Main characteristic of OLSR

- Neighbour sensing: each node periodically broadcasts HELLO messages (it contains list of neighbours, where there exists a valid bidirectional connection + list of the neighbours, which have heard about this node). All one-hop neighbours receive these HELLO packets (TTL is set up to 2)
- MPR Declaration: to build the intra-forwarding database needed for routing packets, each node broadcasts specific control messages called topology control (TC) messages. [18] It contains list of MPR nodes. MPR spread this information within the network through other MPR nodes.
- Routing tables is build up on base of address found in TC messages

2.2.2 Advantages of OLSR

- suitable for widespread topologies
- frequently sending of TC messages as a feature for reducing loops
- Easy to implement QoS - extension to the QoSLR - measurement of important parameters quality services, such as delay, delay variation, bandwidth etc.,

2.2.3 Disadvantages of OSLR

- Impact of using MPR's for small network is minimal
- require relative much storage and performance (maintained information about the entire network)
- routes, which may never be used, are maintained

2.3 ZRP (Zone Routing Protocol)

ZRP, as a hybrid routing protocol, takes the advantages of both worlds. It was designed by Zygmunt Haas of Cornell University to speed up delivery of packets and reduce processing overhead by selecting the most suitable route. ZRP relies on two different protocols: Intra-zone Routing Protocol (IARP) for proactive routing within the zones, Inter-zone Routing Protocol (IERP) for reactive routing outside of the zones and BRP is responsible for forwarding IERP route queries to the peripheral nodes of the broadcasting node. Although the receivers of a broadcast packet are the peripheral nodes, the BRP deliver the query to the IERP at every hop. [19]

2.3.1 Main characteristic of ZRP

- each node individually creates its own zone (contains of the nearest neighbours)
- if destination is in the same zone as the origin → using proactive protocol and its already stored routing table
- if destination is out of the zone → reactive protocol checks each successive zone to find if the destination is located inside that zone. If is the location in that specific zone confirmed it is employed for delivering the packet (on-demand search)
- targeted to large networks

2.3.2 Advantages of ZRP

- finds free loop routes to the destination
- ZRP generate less traffic amount compared to pure proactive or reactive routing
- only local effect when network topology change

2.3.3 Disadvantages of ZRP

- impossibility to resize the zone

3 Ad Hoc problems and threats

In this chapter, we are going to discuss the potential threats/ problems of a mobile ad hoc network (MANET) and explore their current solutions. Access points, base stations or centralized administration is missing, therefore security of ad hoc network becomes inherent weakness. In the end, it will be suggested a solution to mitigate, respectively to eliminate the impact of threats on the topology.

3.1 The Broadcast Storm

Broadcasting is a common process in a network to resolve many issues. Because radio signals are likely to overlap with others in a geographical area, a straightforward broadcasting by flooding is usually very costly and will result in serious redundancy, contention, and collision, to which we refer as the broadcast storm problem. [4] MANET network relies on sending broadcast messages to solve many layer problems, graph-related problems and distributed computing problems as well and each host use CSMA/CA (carrier sense multiple access with collision avoidance) transceiver. Due to host mobility, broadcasting is performed more frequently because of sending the alarm messages, finding the next hop host and paging new hosts.

Attempts for synchronizations in such a network with mobility and broadcast topology information is unavailable. Therefore, broadcasting by flooding seems as the most feasible method. However, we should take into account several complications connected with the MANET broadcasting:

1) multiple physical locations may be covered by the broadcasting of the other hosts as a cause of the omnidirectional propagation
2) many redundant rebroadcasts
3) rebroadcasting hosts, which are too close each other within the topology may cause heavy arguments
4) RTS/CTS (Request to Send / Clear to Send) is inapplicable -> collisions are more likely to show and timing of rebroadcasts is highly correlated [4]

MANET broadcasting is spontaneous and unreliable. It means there is not any mechanism for control of timing and hosts can broadcast message whenever they want. To be off-line, temporarily isolated from the network may lead to the host miss the message and acknowledgements may cause serious medium contention (and thus another “storm”) surrounding the sender, and 100% reliable broadcast is unnecessary in many applications. [4]

CSMAICA network, drawbacks of flooding include:

- **Redundant rebroadcasts**: mobile host rebroadcast a message to its neighbours, however all its neighbours have already received it.
- **Contention**: if many of its neighbours decide to rebroadcast the message, these transmissions (which are all from nearby hosts) may overlay with information already obtained
- **Collision**: the lack of RTS/CTS dialogue and the absence of CD cause higher likelihood of collision more damage. [4]

Following analysis on redundant rebroadcasts show how much costly and inefficient would be using the rebroadcasting. Therefore, it would be used with caution. Let's say we have two points, **A** and **B**, spaced apart **d** distance. Node A wants to broadcast the message and B’s role is to forward the

message. Let be **r** be radii area of A and B, S_A and S_B would be considered as circle areas covered by A's and B's transmissions. We can derive, that the additional area that can benefit from B's rebroadcast is given by following equations.

$$|S| = \pi r^2 - INCT(d) \quad (1)$$

$$INCT(d) = 4 * \int_{d/2}^{r} \sqrt{r^2 - x^2 dx} \quad (2)$$

If d=r, the coverage area $|S_{B-A}|$ is the largest. Rebroadcast can provide only 0~ 61% additional coverage over in compare with the previous transmission.

Let's analyse the case of contention of n=2. Consider the situation, when host **A** broadcast the message and there are hosts **B** and **C** receiving this message. Let B locate at A's area and in order for C to contend with B, it must be situated within the intersection of A's and B's transmission ranges. B. So the probability of contention is

$$/S_{A \cap B}//\pi r^2 \quad (3)$$

Let x be the distance between A and B. Integrating the above formula over the circle of radius x from 0 to r, the expected probability of contention is [4]

$$\int_0^r \frac{2\pi x * INCT(x)/(\pi r^2)}{\pi r^2} \quad (4)$$

In the analysis of collisions, will be explained firstly the back-off procedure, because it's necessary to understand it clearly. At the time when the channel is occupied by transmitting node, the MAC protocol must expand information to other hosts that the media is busy. Back-off mechanism, as a part of MAC protocol, is used to decide the length of node suspension. Selection of back-off mechanism should consider generating back-off timers (adequate time for current transmissions to finish) and avoid unneeded idle time that leads to redundant delay in the network. [7]

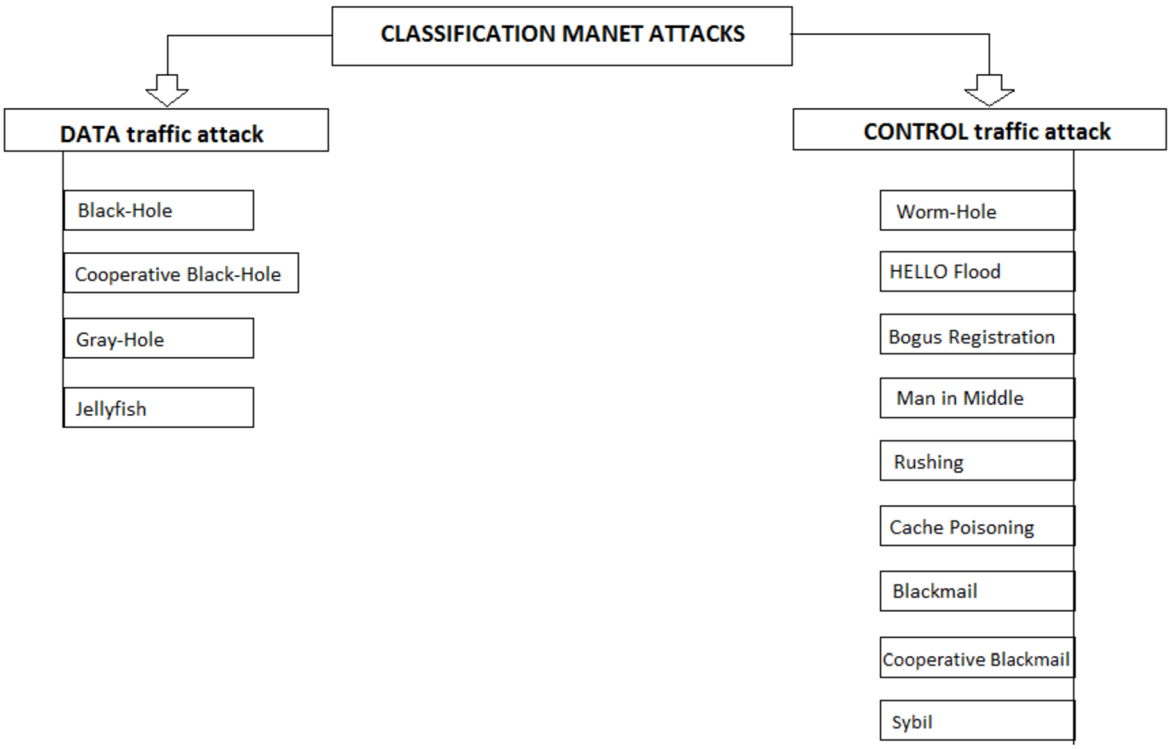
We have to take into account that in MANET networks is no base station or access point. Therefore, we will focus mainly on the behaviour under the distributed coordinate function (DCF). The CSMA/CA mechanism requires a host to start a back-off procedure right after the host transmitted a message, or when a host wants to transmit but the medium is busy and the previous back-off has been done. If the channel clear assessment (CCA) mechanism of the host doesn't capture any activity on the channel during the past slot (a fixed period), the counter is automatically decreased by one. When the counter reaches zero, the back-off procedure is finished. [4]

Let's imagine the scenario with several neighbour hosts hear broadcast from specific host H. There are few options where collisions may occur.

- if in the neighbourhood of host H hasn't occur any broadcasting, host H's neighbours could have mistakenly think there is a free channel. After having passed the DIFS period, they may all start rebroadcasting at around the same time
- because RTS/CTS is missing in a broadcast transmission, damage collisions may have more serious impact.
- Without collision detection (CD), a host will keep sending the packets even if some of foregoing bits have been detected. [4]

3.2 Data Traffic Attacks

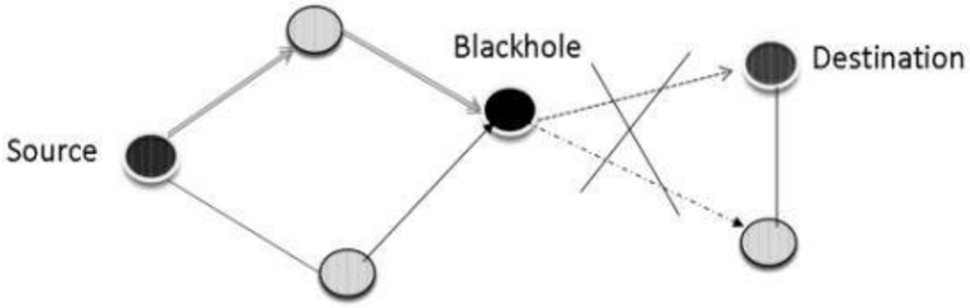
For better clarity, the existing attacks are categorized into two broad categories: DATA traffic and CONTROL traffic attacks. The following groups are based on their characteristics and attacks targets/ goals.



Pict. 2: Classification MANET attacks

3.2.1 Black-Hole Attack

A black hole problem means that malicious node claims itself of being the shortest path to the destination node. However, all packets received by this node are dropped and any of the routing packets are not forwarded to its neighbours. [9] Malicious node replies the RREQ packet sent from source node with a response, which says, that a malicious node has the quickest route to the destination node. As a result, the source starts erroneously sending all of the packets through the black-hole - dropping all data packets passing through it (analogy to black hole in our universe). If the attacking node is a common point of two connecting components of that network, then the network is separated in to two disconnected components. [8]



Pict. 3: Sample of black-hole attack's impact on given topology

Node infected with black-hole virus is a connecting node of two connecting components of that network, then it may effectively divide the topology into two separate disconnected parts as shown above.

Many different strategies to mitigate the impact were developed. Some of them will be described:

(i) Neighbourhood-based and Routing Recovery Scheme [9]

Neighbourhood-based is used as a detection scheme to recognize a malicious node and routing recovery protocol's role is to ensure the correct path. The source node sends the modify route entry control packet to correct (destination) node to rebuild routing path. In this scheme, lower detection time, higher throughput and accurate detection probability are achieved.

(ii) Redundant Route Method and Unique Sequence Number Scheme [9]

As it was mentioned above, a malicious node acts as the quickest and most suitable route to the destination. First proposal, how to eliminate the black-hole is to find more than one route to the destination node. The source sends a RREQ (Route Request) packet to the destination. The receivers, who has route in their routing table to the destination will replies with RREP packet. The sender will stand by until he receives more than 2 RREP packets. He will reply after identifying a safe route. This route will be recognized from the numbers of hops or nodes.

Second proposal, two values are needed to be recorded in two additional tables. One is the last-packet-sequence-numbers for the last packet transmitted to every node and the other is for the last packet received. When any change occurs, these two tables will be updated automatically. Comparing these two table values, the sender node can determine whether there are malicious nodes or not. [9]

As a result, both solutions produce less RREQ and RREP numbers than AODV protocol. Furthermore, solution two is better than solution one due to the sequence number included in every packet in the original routing protocol. Inbound cryptography method can be used to eliminated the communication overhead. [9] However, I must admit that it works for single black attack detection only.

(iii) Time-based Threshold Detection Scheme [9]

According to this method, packet's sequence number will be stored jointly with the received time in a Collect Route Reply Table (CRRT). Then count the timeout based on the arriving time of first route request and decide if the path is valid or not according to above threshold value.

3.2.2 Cooperative Black-Hole Attack

Detect the cooperative black-hole attack is incomparable more demanding then mechanism for black-hole detection. Malicious nodes collaborate to convince the other nodes about the authenticity of their fabricated routing information. [9]

(i) DRI Table and Cross Checking Scheme [9]

Every node must carry one more extra DRI table, 1 stands for true and 0 for false. The entry is composed of two bits- "From" (from specific node) and "Through" (though the node) which stands for information on routing data packet. [9]

Procedure starts when source node sends the RREQ packets to each node, which replies the RREP packet. The intermediate node (IN) transmits next hop node (NHN) and DRI table to the SN, then the SN cross checks its own table and the received DRI table to determine the IN's honesty. After that, SN sends the further request to IN's NHN for asking its routing information, including the current NHN, the NHN's DRI table and its own DRI table. [9] Subsequently, the source node compares the received information by cross checking and decide if the given node is trustful or not. However, it wastes 5 to 8% communication overhead, and slightly increases the packet loss percentage because of the secure route discovery delay [9], but it's almost 50% better than other solutions

(ii) Distributed Cooperative Mechanism (DCM) [9]

Distributed Cooperative Mechanism consist of four sub-modules:

- local data collection =>
- local detection =>
- cooperative detection =>
- global reaction.

In the first phase, local data collection, each node in the network create and maintain estimation table. Nodes evaluates also the information of transmitting packets to clarify whether there exists any suspicious node. If there is a suspected existence, the detect node initiates the local detection phase to recognize possible black hole. Cooperative node receives a check packet from initial detection. If the inspection value is positive, the controlled node is marked as a normal node. Otherwise the cooperative detection procedure is initialized by detection node and deals with broadcasting and notifying all one-hop neighbours to cooperate in the decision making. Finally, the global reaction phase set up a notification system and sends warning messages network if needed. [9] Each node carries its own black hole list and transmission route in other mode.

Delivery ratio ranges between 64.12 to 92.93% [9] when using different threshold values.

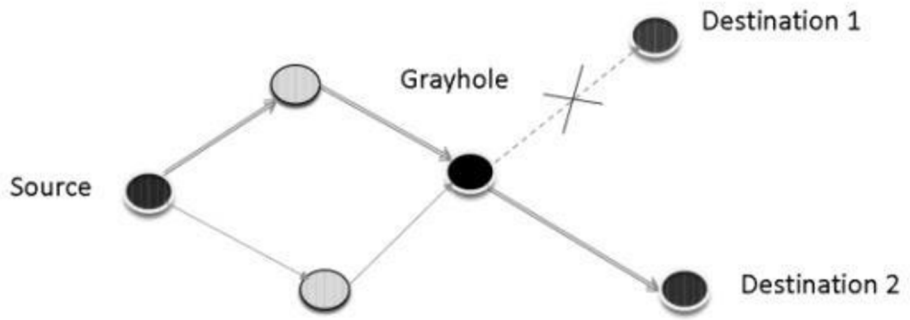
(iii) Hashed-based MAC and Hash-based PRF Scheme [9]

These two proposals (message authentication code (MAC) and the pseudo random function (PRF)) are used to prevent cooperative black-holes, seek collaborative malicious nodes and to provide fast message verification and group identification.

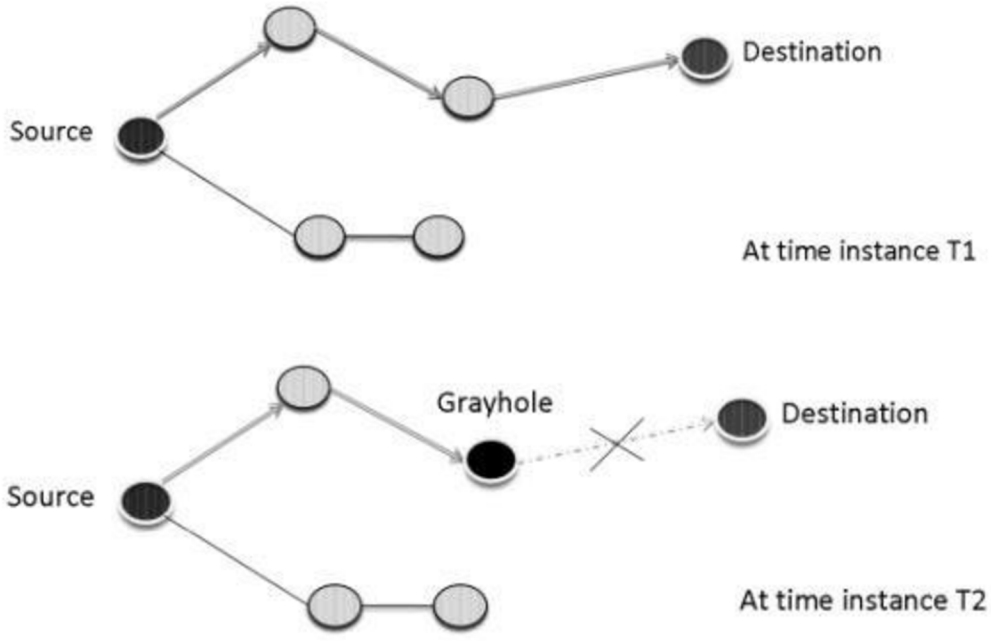
Each node obtain a secret key K_i, and $K_i = G_k(ri)$. The sharing key Ki is secret (hide from others), hence, ri is a random number and repeatedly applying PRF on ri by k times. When source node receives a packet, it checks Ki-d to find whether the key used for the MAC is disclosed or not. If ki is disclosed -> checks the MAC. [9]. If above two conditions will be passed, this packet would be considered as a secure.

3.2.3 Gray-hole Attack

Instead of blindly dropping the all data packets as in Black-hole attack, the attacker selectively drops the packets originating from a single IP address/range of IP addresses and forwards the remaining packets (node dependent attack) or drops packets based on some predetermined/trigger time (time dependent attack).



Pict. 4: Sample of node dependent attack



Pict. 5: Sample of time dependent attack

Gray-hole has two major phases

1. the malicious node accomplishes the AODV protocol to claim that it has the best valid route to destination node, with the intension of including or confusing the packets, even though route is fake
2. the malicious nodes drop the irregular packets with a certain vision

(i) Technique to detect grey-hole using DRI table [10]

Initiator Node (IN) initiates the suspected node recognition procedure and first chooses a Cooperative Node (CN) in its region, based on its DRI records and broadcasts a RREQ message to only its 1-hop neighbours requesting for a route. The IN will receive a number of RREP messages from its neighbours. One of the replies will be definitely a RREP message from the Suspected Node (SN). As RREP is received from the SN, the IN sends a query packet to the CN through the SN. After the TTL value of the query packet is over, the IN checks the CN whether it has received the query packet or not. If it answers in positive way, then its DRI table is modified by IN. However, if the query packet doesn't reach the CN, the IN increases its level of suspicion about the SN and starts the suspicious node recognition procedure. [10]

(ii) Detection using credit based technique [10]

New algorithm was developed known as Credit Based AODV (CBAODV) and is capable to find out existence of cooperative grey-hole nodes. Firstly, for every node is assigned a permanent value as the neighbour credit value. This credit value is moveable, it increases when it receives a route request packet (RREQ) and decreases when it receives the route reply (RREP) packet. If this value falls below 0 (negative numbers) for one of the neighbour, it seems as the grey-hole attacker. As a next step, all current established paths are removed from its routing table which is going through that node. [10]

3.3 Control Traffic Attacks

In this chapter, we are going dealing with the right part of graph shown on picture 2. These attacks stand and fall by the fact that node, for successful attack, should be part of the network. Malicious node can easily disrupt the network by hijacking the routing tables, modifying the valid routes or in worse case, established itself as shortest route to any destination by utilizing the unsecure routing protocols. Using these kind of unsecure protocols is highly un-recommended by many network experts.

3.3.1 HELLO flood

Some routing protocols require the nodes to announce themselves by broadcasting hello messages. If some node reaches this message it means that node is located within radio range of the sender. However, attacker can easily extend radio range by large enough transmission power and convince every other node in the network that the attacker is its neighbour. To initialize sending hello flood attacks packets node can simply rebroadcast overhead packets with enough power to be received by every other node in the network. [20]

(i) Signal strength technique [20]

Each node checks the signal strength of the received hello messages with known radio range strength. If the ratio is the same, then sender node is classified as a "friend". Otherwise it is classified as a "stranger". If the node appears as a "stranger", technique called "client puzzles" is used to detect its validity, possible threats respectively (this technique will be discussed in next point (ii)). [20]

Some primary assumption are-

- Communication stays within fixed radio range.
- Transmitting and receiving signal strength of all sensor nodes are the same
- All sensor nodes are homogeneous (same HW and SW, battery power etc.).
- Every sensor node knows the fixed signal strength used in its communication range.
- A time threshold is used, which denotes the expected time of reply message.
- Hello message counter has been used by all sensors to keep the record of number of hello requests received in an allotted time. [20]

$$Pr = (Pt*Gt*Gr*Ht^2*Hr^2)/(d^4*L) \quad (5)$$

Equation to calculate strength of the signal where *Pr* is received signal power (in watts), *Pt* is transmission power (in watts), *Gt* is the transmission antenna gain, *Gr* is the receiver antenna gain, *Ht* is the transmitter antenna height (in meter) and *Hr* is the receiving antenna height (in meter), *d* is the distance between transmitter and receiver (in meter), and *L* is the system loss (a constant). [20]

According to result from equation (5) shown above, the node is decided if is sender friend or not by simple following rule. The sender is considering as a friend as long as his fixed strength signal is equal to signal strength. However, if situation occur when strength signal is greater than fixed signal strength, the sender is labelled as a "stranger". There may be a third option- the signal of received hello packet is approximately same, but no equal. To differentiate between a statement "friend" and "stranger" helps technique called "client puzzle".

The idea of hello message based client puzzles scheme (MBCP) is that the larger the number of hello messages sent -> more difficult puzzles to solve. Each node counts number of received hello packets of allocated time and a puzzle generating capability. [20]

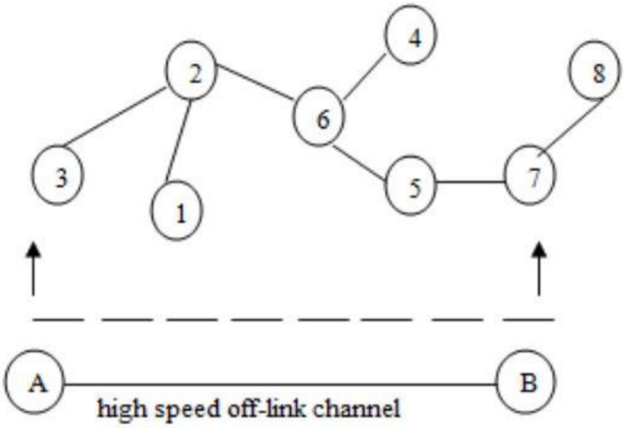
(ii) REHIDAN algorithm to identify flooding attacker nodes [21]

The Real-time Host Intrusion Detection for Ad hoc Networks (REHIDAN) algorithm was created to minimize the effectiveness of the attacks. The REHIDAN algorithm uses the idea of neighbour suppression algorithm isolating through which, the attacker is isolated from the neighbour nodes. It is implemented, with OPNET. [21]

3.3.2 Wormhole Attack

Base of this attack consist of two attacker nodes connected to each other via link. For clearly understanding, let's name it as a tunnel. The both attacker node and wormhole nodes claims in the network a route that is shorter than the original one. [22] But this is not condition- wormhole fake path may be shorter as well.

Let's take an example- we want to send packet over following topology from source node 3 to destination node, which represents node number 7. The legitimate path would be path over nodes 2, 6 and 5. However, after activation of malicious nodes, which create illusion shortest path to the destination, they will be able to catch and hear all of the communication.



Pict. 6: Sample *of topology with indicated tunnel*

The wormhole attack can be launched by modes listed below as [22]:

- **Packet Encapsulation or In-band Channel** –malicious node capture all of the packets from genuine nodes, adjust header and transmit them to other attacker nodes. It can also drop the packet or forward to other nodes
- **Out of Band Channel –** connection between malicious node ensure the outer link with high bandwidth
- **High Power Transmission –** attacker convince other nodes because he captures the packets and transmit them with higher transmission power. Whole traffic will pass through this link and follow it, because legitimate nodes will consider this link as trustful

We can classify three kind of wormhole attack according to [22]:

- **Open Wormhole Attack** identity of malicious nodes are included in header of packets and legitimate nodes are aware of the presence of the wormhole nodes. However, wormhole nodes may not be necessary be malicious and both attacker nodes are visible
- **Closed Wormhole Attack** attacker doesn't update info in header at the time of discover process, legitimate nodes are not aware of their presence. Packets are transmitting via tunnel and then either forward the packet or discard; here are attackers invisible for neighbours
- **Half Open Wormhole Attack** this is combination of 2 mentioned option listed above- one of them is invisible and second one visible + it updates its identity in the packets

Detection technique of wormhole attacks:

(i) Candidate Loop Selection [23]

After the shortest path tree is established, each node knows its shortest paths to the root node. Changes of shortest paths are exchanged between the neighboring nodes. These nodes establish "cut pairs". Two shortest paths establish a loop which is consider a candidate loop. Decision threshold is set up on the length of this loop. The threshold is related to span of wormhole attacks. If we want to detect all wormholes across *h* hop span, the threshold should be set up to *h* hops [23]

(ii) Geographical leashes & temporal leashes [8]

A leash is added to each packet in order to restrict the distance the packets are allowed to travel. A leash is associated with each hop. Thus, each transmission of a packet requires a new leash. A geographical leash is intended to limit the distance between the transmitter and the receiver of a packet. A temporal leash provides an upper bound on the lifetime of a packet.

3.3.3 Rushing Attack

A rushing attacker results in denial of services and thanks to duplicate suppression mechanism quickly forwards route discovery packets to obtain access to the forwarding group. [24] When sender broadcast a RREQ (route request) message in neighbourhood and valid routes replies with RREP (route reply) contained route info. Suppression mechanism might be applied to decrease the number of route requests and replies chatter. Rushing attacker quickly forwards with a malicious RREP on behalf of some other node skipping any proper processing. Actual valid RREP message from valid node will be discarded and consequently the attacking node becomes part of the route. In rushing attack, attacker node does send packets to proper node after its own filtering is done. So as a result, networks from outside seems to behave normally. [8]

(i) Secure neighbour detection and secure route discovery procedure [24]

Generally, nodes broadcast message (Hello) to allow its neighbour to detect them. Secure neighbour detection relies on bidirectional link between two nodes. In those on-demand protocols, nodes who receive a route request consider itself the neighbour of previous-hop node. When a node transmit a request is claim a path between sender and receiver, but this secure neighbour detection

cannot prevent an attacker to receiving a request. In the case, that previous node hasn't been authorized and attacker took control over this node, it can claim to be any node propagating a request. Next hop node will trust this statement automatically. To prevent unsecure route discovery a randomized path selection technique is used.

In tradition way of route request, when the receiving node receive request it immediately forward the request. However, in modified approach, receiving node is waiting to collect all route requests, select some requests randomly and consequently chosen request will be forwarded.

(ii) The concept of threshold [24]

Typical behaviour of attacker in the network is quickly forward RR packet or increase the transmission speed of packet. This cause that receiver receives this fast packet as first and legitimate RR packets will be discarded. Therefore, threshold value was invented, which says packets should be delivered to neighbour destination at fixed time interval. As a result, rushed packet reaches the node earlier than it's given by threshold value and sender is labelled as an attacker.

4 Simulation in NS-3

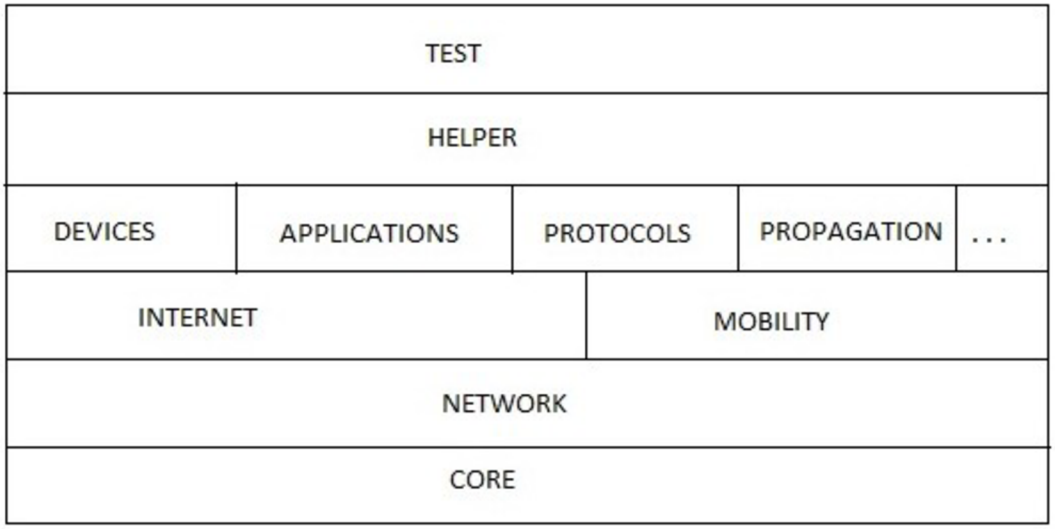
In this chapter, it will be shown how to create simple MANET network with several nodes and default setting. It will be explained the work of particular helpers, an impact of their settings on the topology.

4.1 NS-3 simulator

NS-3 is a discrete-event network simulator in which the simulation core and models are implemented in C++. [5] It focused primarily on research and teaching. It is available for free and publicly available under the GNU GPLv2 license for research, development and use. First version (ns-1) of NS simulator was released in 1995 and derived from an earlier simulator known as REAL by S. Keshav. NS-3 is not a backwards-compatible extension of NS-2. Infrastructure of NS-3 allows to simulate such as models, which should be aligned with the simulations needs of modern networking research. It is used to study system behaviour in a highly controlled, reproducible environment, and to learn about how networks work. Users will note that the available model set in ns-3 focuses on modelling how Internet protocols and networks work, but ns-3 is not limited to Internet systems. [3]

Many simulations tools exist for exploring of network behaviour or protocols. I would like to highlight some distinguishing features in compare with other tools:

- Using C++ programming language and its libraries that can combined together also with external software libraries
- In this version of the software, integrated graphical user interface environment in which all tasks are indicated
- NS-3 offers several external animators and data analysis and visualization tools
- Ns-3 is running primarily on Linux systems, although some alternatives exist. Such as FreeBSD, Cygwin (for Windows) and native Windows Visual Studio (it is in the process of being developed)



Pict. 7: Hierarchical structure of NS-3 simulator

4.2 Creating simple program

Ns-3 is built as a system of software libraries work together. User programs can be written that links with (or imports from) these libraries. User programs are written in either the C++ or Python programming languages. Base of the program contains from the pre-built libraries, but at present, many users actually do their simulations by editing ns-3 itself. Because in some cases, pre-built libraries do not offer needed headers files. Whole program is allocated within the namespace (to distinguish classes with the same name), in this case *ns3.*

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

using namespace ns3;
```

In the next lines of the script are defined types of logging massages, which allows debug logging at the different levels. As a result, will be messages printing out by nodes, devices, protocols or applications as packets are coming through the topology and there exist several levels how deep we want to go.

```
LogComponentEnableAll (LOG_LEVEL_WARN);
LogComponentEnable ("AodvRoutingProtocol", LOG_LEVEL_WARN);
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
```

- *NS_LOG_ERROR* – error messages,
- *NS_LOG_WARN* – warning messages,
- *NS_LOG_DEBUG* – relatively rare, ad-hoc debugging messages,
- *NS_LOG_INFO* – informational messages about program progress,
- *NS_LOG_FUNCTION* – a message describing each function called,
- *NS_LOG_LOGIC* – messages describing logical flow within a function,
- *NS_LOG_ALL* – everything,

We also provide an unconditional logging level that is always displayed, irrespective of logging levels or component selection [1]

- *NS_LOG_UNCOND* – Log the associated message unconditionally.

```
NodeContainer nodes;
nodes.Create (8);

MobilityHelper mobility
mobility.SetPositionAllocator("ns3::GridPositionAllocator",
        "MinX", DoubleValue (20.0),
        "MinY", DoubleValue (100.0),
        "DeltaX", DoubleValue(100),
        "DeltaY", DoubleValue (100),
        "GridWidth", UintegerValue(3),
        "LayoutType", StringValue ("RowFirst"));

 mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
 mobility.Install (nodes);
```

Next two lines of code will actually create the ns3- nodes. Using the *NodeContainer* helper is the simplest way how to add nodes to our topology. However, the previous command creates only empty container. Therefore, we must fulfil it by adding two nodes. In order to wireless communication works properly, we should follow the rules about nodes layout and set up them in a smaller distance. Otherwise, a compilator could throws error. Using following attributes, we set the size of the grid.

- *GridWidth* : The number of objects layed out on a line
- *MinX,MinY* : The x/y coordinate where the grid starts
- *DeltaX, DeltaY* : The x/y space between objects
- *LayoutType* : The type of layout [2]

Applying the mobility model is not goal, therefore we have arranged the nodes on constant positions using *ConstantPositionMobilityModel* command. *NetdeviceContainer* topology helper is needed for installing the wireless technology on the devices.

```
NetDeviceContainer devices;

NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifiMac.SetType ("ns3::AdhocWifiMac");
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper :: Default();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper :: Default ();
wifiPhy.SetChannel (wifiChannel.Create ());

WifiHelper wifi = WifiHelper :: Default ();
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
        "OfdmRate6Mbps"), "RtsCtsThreshold", UintegerValue (0));
devices = wifi.Install (wifiPhy, wifiMac,nodes);
```

With *NqosWifiMacHelper, YansWifiPhyHelper, YansWifiChannelHelper* we adjust necessary parameters for properly working of *WifiHelper*. Once the PHY and MAC helpers are configured, channel is going to be created by command *Install* with attributes given from previous helpers. The *SetRemoteStationManager* method tells the helper the type of rate control algorithm to use [3].

```
AodvHelper Aodv;
InternetStackHelper stack;
stack.SetRoutingHelper (Aodv);
stack.Install (nodes);
```

We will use InternetStackHelper to install the chosen protocol for ad hoc networks- AODV and subsequently, apply it on the nodes.

```
Ipv4InterfaceContainer interfaces ;

Ipv4AddressHelper address;
address.SetBase ("192.168.0.0", "255.255.255.0");
interfaces = address.Assign (devices);
```

Here is shown using command *Ipv4AddressHelper* to assign IP address to device ‘s interfaces. Firstly, we initialize an address range. Then the interfaces of devices will be associated with IP addresses from network number 192.168.0.0 in this case, as seen above. System does it automatically and there is no option how to assign specific address on chosen interface. By default, the address allocation starts at the lowest possible address and increase monotonically.

```
UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (nodes.Get (0));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (0), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (10));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps1 = echoClient.Install (nodes.Get (9));
clientApps1.Start (Seconds (2.0));
clientApps1.Stop (Seconds (10.0));
```

One of the core abstractions of the ns-3 system is the Application. In this script we use two specializations of the core ns-3 class Application called *UdpEchoServerApplication* and *UdpEchoClientApplication* [3]. They are used to generate and echo simulated network packets. With this combination of attributes, echo client sends one 1024- bytes packet every 1 second to port 9. IP address corresponds to zeroth interface in the nodes container. Then, it is needed to install, both, server and client on the specific nodes.

```
AnimationInterface anim ("mojProjektVizualizacia.xml");
Simulator::Stop (Seconds(10));

Simulator::Run ();
Simulator::Destroy ();
return 0;
```

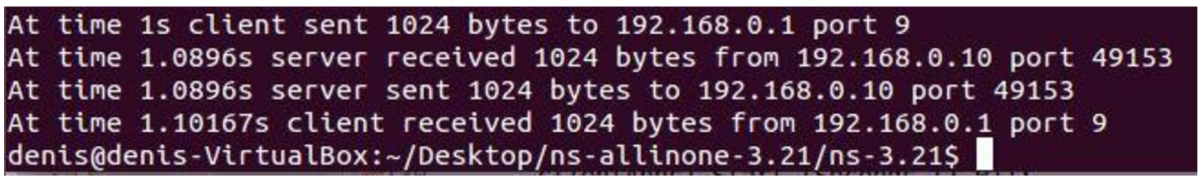
AnimationInterface command (available as a standard in NS-3 with ns-3.20 version or higher) allows us animate packets, view the statistics, filtering the traffic data, etc. It will be more explained in the further chapters. When *Simulator::Run()* is called, the system will begin looking through the list of scheduled events and executing them. For instance, scheduled starts events, which will enable the echo server application and client application. We have to make sure, that definition for stopping the simulation is written before the run command. Otherwise, it may happen that the simulation will never end. Cleaning up the remains (destroying all of the objects that were created) is done by calling the function *Simulator::Destroy().*

For building the scripts exists many tools, the most widespread is *make*. However, It's use is not recommended, especially in complex and highly configurable topologies. Therefore, NS-3 has developed an alternative- waf (based on Python language). Type the following command in the terminal:

```
./waf --run scratch/first
```

4.3 Gaining results and topology

In the first lines of our code are declared logging messages, which enable us to discover the results. In our example, *UdpEchoClientApplication* and *UdpEchoServerApplication* are used, therefore we set up logging message on the level INFO.



Pict. 8: Output shown in terminal after running the program

As we can see, logging message on the echo client indicate that it has sent 1024 bytes' packet to the address 192.168.0.1 assigned to the server on the port 9. Attributes, as a size of packets and number of the destination port were chosen by us, but assigning the address is automatic process and we cannot influence that. Server echoes the packet to the address 192.168.0.10 on the port 49153 and you see the echo client log that it has received its packet back from the server. Whole communication took little more than 0.1 second.

Next option how to examine the results is an implementation of trace files. The ns-3 helper *enablePcap* is used to create these files in .pcap format. Output can be written by the most popular traffic analyser- Wireshark (formerly called Ethereal) with options as a graphical user interface, used protocols or simply by using tcpdump.

Capturing the packets info will be enabled by typing the following command:

```
wifiPhy.EnablePcapAll("myFirst");
```

The pcap helper will actually create files named „myFirst-0-0.pcap“, and „myFirst-1-0.pcap“ located in the *scratch* file. First zero represents the number of node and second zero is a device. For reading output with tcpdumb, what is actually the easiest way to look the pcap file, type to terminal:

```
$ tcpdump -nn -tt -r myFirst-0-0.pcap
```

Next option how to trace packets is printing out the routing tables. As a result of following commands is the file blackHole.routes located in the *scratch* file with tabular and detailed information such as destination IP address, gateway, outgoing interface, flags, expire time and number of hops for every node separately.

```
Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper>
("blackHole.routes", std::ios::out);

Aodv.PrintRoutingTableAllAt (Seconds (4), routingStream);
```

As a sample was chosen the routing table of node 1 from blackhole topology at the time 2 seconds. As we can see, there are saved only those destination IP addresses nodes that were used for routing.

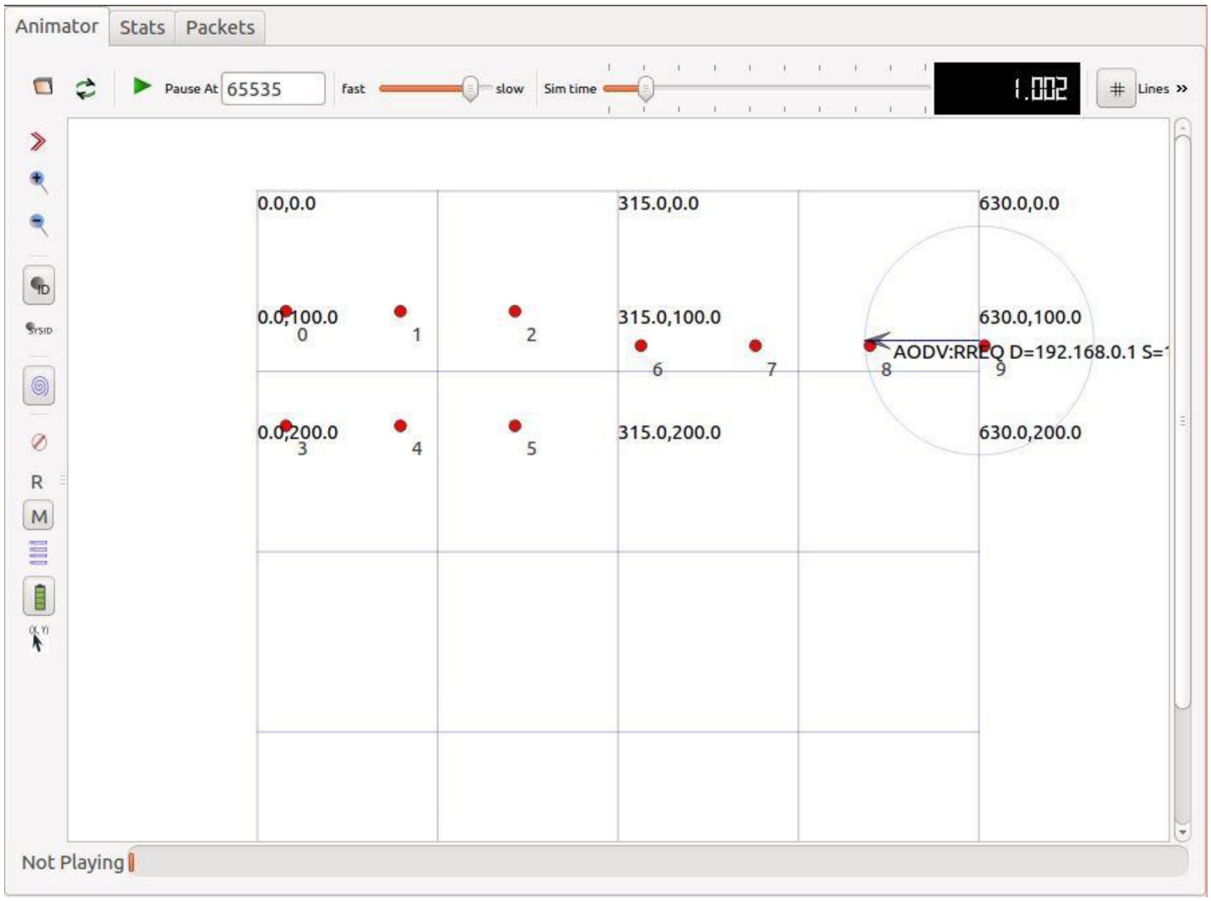
```
Node: 1 Time: 2.00s
AODV Routing table
Destination     Gateway         Interface       Flag    Expire          Hops
127.0.0.1       127.0.0.1       127.0.0.1       UP      9223372034.85   1
192.168.0.1     192.168.0.1     192.168.0.2     UP      10.23           1
192.168.0.3     192.168.0.3     192.168.0.2     UP      2.09            1
192.168.0.5     192.168.0.5     192.168.0.2     UP      2.04            1
192.168.0.10    192.168.0.3     192.168.0.2     UP      4.22            5
192.168.0.255   192.168.0.255   192.168.0.2     UP      9223372034.85   1
```

Pict. 9: Routing table after running sample program

As was mentioned above, for visualization is employed by software NetAnim (Network Animator), which works with .xml files. After the successful launch of the application, a file named "mojProjektVizualizacia.xml" will be created. For calling the NetAnim program is used the following command. Sample of graphical user interface is shown on the picture number 2.

```
denis@denis-VirtualBox:~/Desktop/ns-allinone-3.21/netanim-3.105$ ./NetAnim
```

For demonstration will be used the topology corresponding to actual topology for simulating the black-hole attack.



Pict. 10: Visualization of sample topology

The topology consists of 10 nodes, with following attributes:

- nodes are divided into one rectangle 2x3 and one line of 4 nodes
- they are 100 metres apart from each other, in the vertical and horizontal way
- node 1 represents the server and node 9 represents echo client
- static positions, no motion model wasn't applied

I have chosen this layout because of only one way how to get from server to the destination. That is necessary for observation the behaviour of the black-hole and the grey-hole attacks where only one possible way is needed.

5 Threats effects

In this chapter are listed several examples of attacks in MANET network and their influence on continuity of flowing of the packets.

5.1 Black-hole attack

5.1.1 Black-Hole implementation

Black-hole attack belongs to the group of basic assaults based on influencing data traffic. In this demonstration will be implemented a solution, when we set down the node number 7 in the time 4 seconds.

```
if (BH)
 {
        Ptr<Node> nod = nodes.Get (7);
        Ptr<Ipv4> nmbNode = nod->GetObject<Ipv4> ();
        uint32_t ipv4ifIndex = 1;
        Simulator::Schedule (Seconds (4), &Ipv4::SetDown, nmbNode, ipv4ifIndex);
}
```

Calling this extension is made by command prompt with following command:

```
denis@denis-VirtualBox:~/Desktop/ns-allinone-3.21/ns-3.21$ ./waf --run "scratch/
blackHole --BH"
```

Pict. 11: Calling BH extension

For calling the function *SetDown* is needed an attribute such as a number of interfaces. However, we must associate node to the object, because the function *Simulator::Schedule* requests the objects. Firstly, we create a pointer for the node number 7 called *nod* and subsequently, we create next pointer to the object of ipv4 interface called *nmbNode*. The *ipv4ifIndex* command is required for the function *Simulator::Schedule*, but I haven't found the purpose of that. The first ifIndex is 0 for loopback, then the first p2p is numbered 1, then the next p2p is numbered 2.

```
At time 3s client sent 1024 bytes to 192.168.0.1 port 9
At time 3.0101s server received 1024 bytes from 192.168.0.10 port 49153
At time 3.0101s server sent 1024 bytes to 192.168.0.10 port 49153
At time 3.02064s client received 1024 bytes from 192.168.0.1 port 9
At time 4s client sent 1024 bytes to 192.168.0.1 port 9
At time 5s client sent 1024 bytes to 192.168.0.1 port 9
```

Pict. 12: Output in terminal after running black-hole attack

As we can see, the bidirectional communication between the server and the client works properly until 4 second. Client and server sent and received packets without any changes. However, after the activation of a malicious node in the time 4 second every packets, which were sent to this node, were dropped. The client is trying to reach the server every one second, but the packets will never get further than to the node 7.

As is shown on the picture below, after the update the routing tables at the time 9 seconds (AODV routing protocol rely on sending HELLO messages every few seconds), nodes, located on the left from the malicious node, realized that interfaces, which they were trying to reach, were unavailable. Therefore, they set up corresponding flags to state "down".

As a sample have been chosen the node 6's routing tables with IP address 192.168.0.7 before update the routes and after.

- *destination* – destination IP address, IP address 192.168.0.255 serves as a broadcast address to explore neighbours or to find a best path
- *gateway* – next hop IP address
- *interface* – IP address of given node (IP address 127.0.0.1 serves as a loopback address for testing the functionality of the interface, it has no hardware associated with it and it isn't physically connected to a network)
- *flag* – availability of the destination node
- *expire* – remaining time (in seconds) to resend HELLO packets
- *hops* – number of nodes to reach a given destination node

In the following pictures, we can observe the difference between the routing tables in 6. second and 7. second. Shortly after the expiration of 6 seconds the time for resend HELLO message expires and routing tables are updated.

```
Node: 6 Time: 6.00s
AODV Routing table
Destination     Gateway         Interface       Flag    Expire          Hops
127.0.0.1       127.0.0.1       127.0.0.1       UP      9223372030.85   1
192.168.0.1     192.168.0.3     192.168.0.7     UP      6.24            3
192.168.0.3     192.168.0.3     192.168.0.7     UP      1.02            1
192.168.0.8     192.168.0.8     192.168.0.7     UP      0.02            1
192.168.0.10    192.168.0.8     192.168.0.7     UP      0.37            3
192.168.0.255   192.168.0.255   192.168.0.7     UP      9223372030.85   1
```

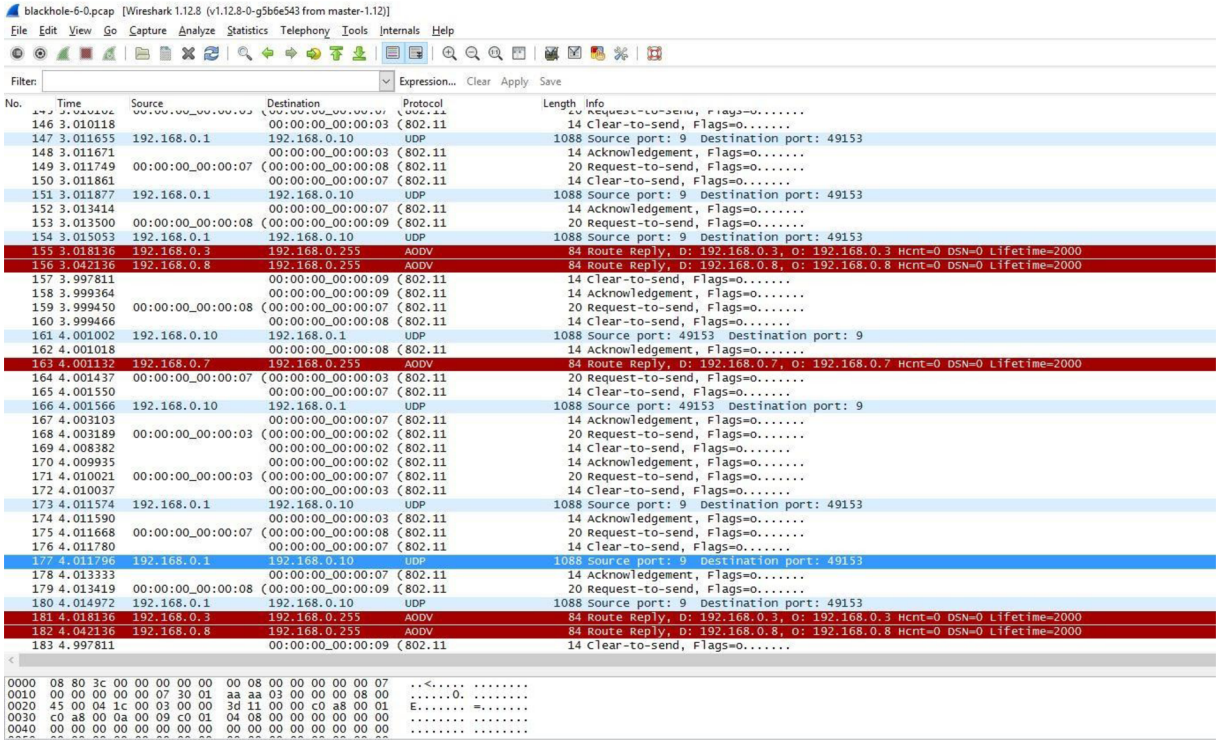
Pict. 13: Node 6's routing table in time 6 sec. after running black-hole attack

```
Node: 6 Time: 7.00s
AODV Routing table
Destination     Gateway         Interface       Flag    Expire          Hops
127.0.0.1       127.0.0.1       127.0.0.1       UP      9223372029.85   1
192.168.0.1     192.168.0.3     192.168.0.7     UP      5.24            3
192.168.0.3     192.168.0.3     192.168.0.7     UP      2.05            1
192.168.0.8     192.168.0.8     192.168.0.7     DOWN    14.02           1
192.168.0.10    192.168.0.8     192.168.0.7     DOWN    14.04           3
192.168.0.255   192.168.0.255   192.168.0.7     UP      9223372029.85   1
```

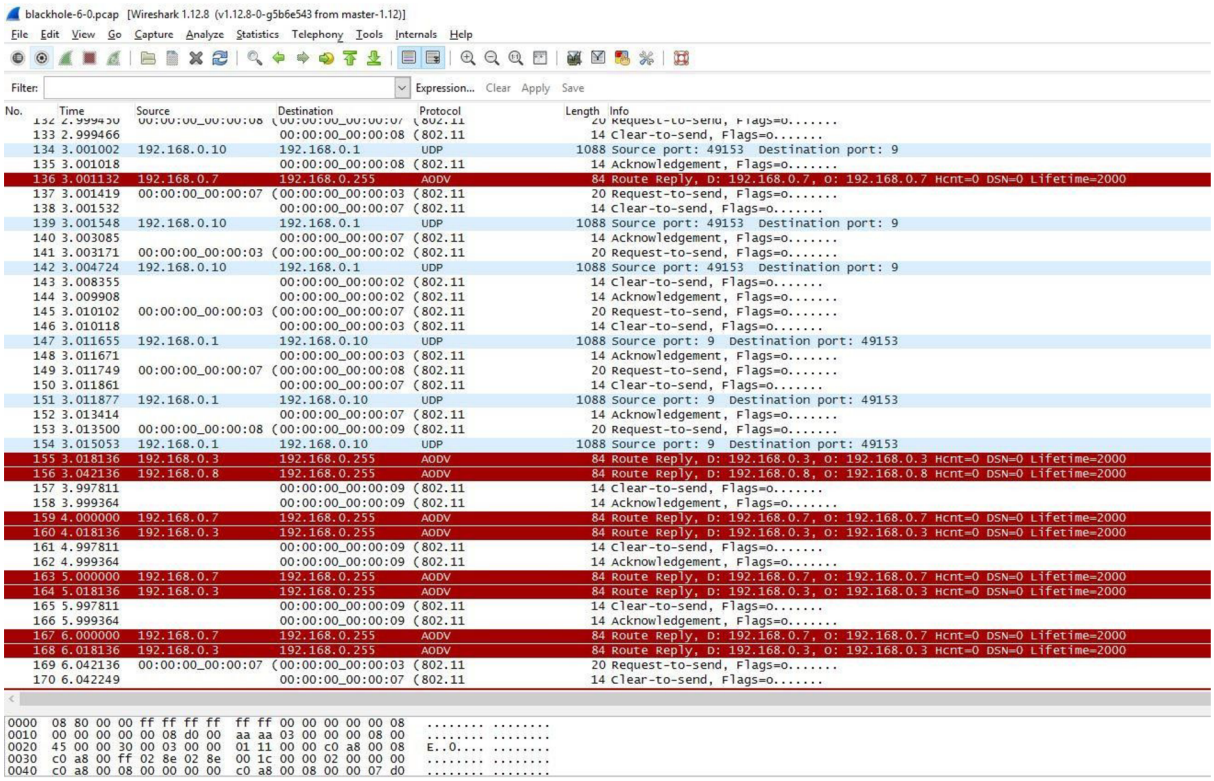
Pict. 14: Node 6's routing table in time 7 sec. after running black-hole attack

The widespread software Shark will serve us for another sample of black-hole network attack impact. I have chosen this SW for decoding the PCAP files generated by every node. In network terminology, PCAP originated as an abbreviation from "**p**acket **cap**ture". In the field of computer network administration, PCAP consists of an application programming interface (API) for capturing

network traffic. Unix-like systems implement PCAP in the libpcap library; Windows uses a port of libpcap known as WinPcap. Monitoring software may use libpcap and/or WinPcap to capture packets travelling over a network. [28]



Pict. 15: Node 6's PCAP file without active black-hole attack



Pict. 16: Node 6's PCAP file with active black-hole attack in 4. sec.

As we can see on the attached pictures above of PCAP files of node num. 6 the differences are quite obvious. Without active black-hole attack, the node number 0 is forwarding UDP packets to the destination node to the port 9 at interval every one second (as was set in the source code).

After activation BH in the time 4 seconds, we can observe destination non availability- AODV protocol is searching and trying to reach destination via alternative route, but without success.

5.1.2 Mitigating the impact of the black-hole

Firstly, I would recommend to find more than 1 path to the destination, however in our case, there doesn't exist any redundant way how to reach destination node. Therefore, I would select the last-packet-sequence-number solution, which was mentioned above. It's needed just carry two more values in the tables for comparison. From my point of view, it seems to be simplest solution in regard to the given topology. It produces less service messages (RREP and RREQ) than AODV protocol and communication overhead can be reduced by proper the inbound cryptography method.

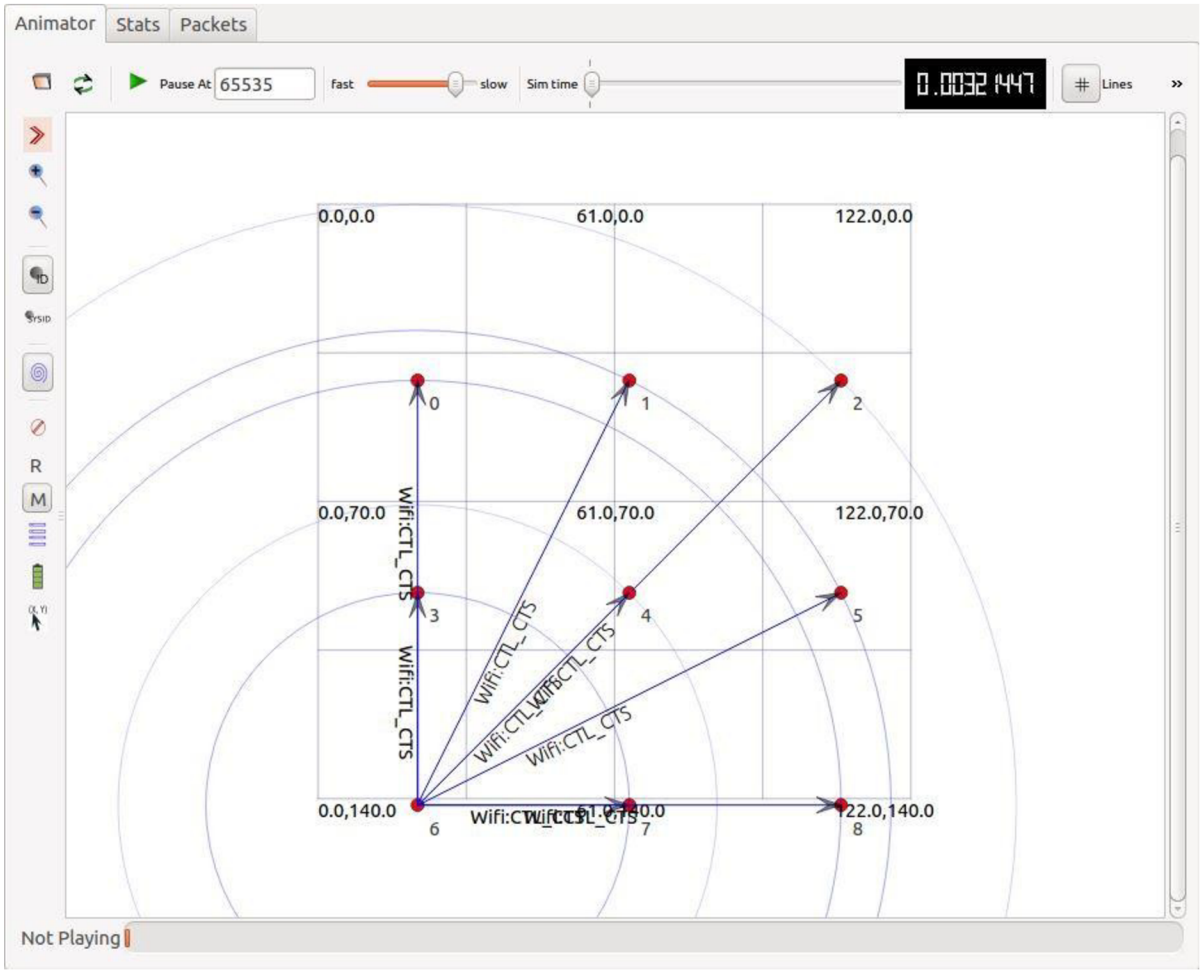
5.2 Grayhole attack

5.2.1 Grayhole implementation

For simulating gray-hole attack was used different layout scheme described by following “mobility” code:

```
mobility.SetPositionAllocator("ns3::GridPositionAllocator",
        "MinX", DoubleValue (22.0),
        "MinY", DoubleValue (40.0),
        "DeltaX", DoubleValue(50),
        "DeltaY", DoubleValue (50),
        "GridWidth", UintegerValue(3),
        "LayoutType", StringValue ("RowFirst"));
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
```

This code leads to graphical output as shown below.



Pict. 17: Topology used in gray-hole attack

Grayhole could be implemented in two ways:

- Node dependent attack - drops DATA packets with specific source or destination address, while for other nodes it behaves normally. Data packets are transmitting to the destination nodes correctly
- Time dependent attack - drops DATA packets based on some predetermined time (in first minute acts normally and in second minute drops everything) while behaving normally during indefinite time

I have chosen the node dependent attack (because of the more complicated implementation in compare to time dependent attack), which is executed by following code:

```
TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
Ptr<Socket> receiver = Socket::CreateSocket (c.Get (5), tid);
InetSocketAddress local = InetSocketAddress ( ipInterfaces.GetAddress(5), 80);
receiver->Bind (local);

Ptr<Socket> middle = Socket::CreateSocket (c.Get (4), tid);
InetSocketAddress mid = InetSocketAddress ( ipInterfaces.GetAddress(4), 80);
middle->Bind (mid);
```

There are certain factory objects that can create sockets. Each factory is capable of creating one type of socket, and if sockets of a particular type are able to be created on a given node, then a factory that can create such sockets must be aggregated to the node *c.Get(5)*.[25] Class *TypeId* was called as a unique identifier for an interface. This class records a lot of meta-information about a subclass of the object base class as the base class of the subclass, the set of accessible constructors in the subclass or the set of 'attributes' accessible in the subclass. Examples of TypeIds to pass to this method are *ns3::TcpSocketFactory, ns3::PacketSocketFactory,* and *ns3::UdpSocketFactory.*

In following lines were established smart pointers to sockets objects called “receiver” and “middle”, which create sockets on the interface number 5: *c.Get(5)* and number 4 (this refers to the nodes nr. 5 and 4 in the layout above). *InetSocketAddress* class holds an Ipv4Address and a port number to form an ipv4 transport endpoint. In our case, the both receiver nodes works on port 80 and IP address is obtained by calling function *GetAddress()* with number of interface over the list *ipInterface.*

Sending a socket through the network require these following steps [26]:

- Bind() - allocate a local IPv4 endpoint for this socket
- Connect() - initiate a connection to a remote host
- Send()- send data (or dummy data) to the remote host
- Recv()- read data from the socket (on the remote side)
- Close()- close a socket

```
Ptr<Socket> source=Socket::CreateSocket(c.Get(0),tid);
Ptr<Socket> source2=Socket::CreateSocket(c.Get(6),tid);
source  ->Connect (mid);
source2->Connect (mid);
```

Smart socket pointer “source” and “source2” are established on the node nr. 0 and nr. 6 and subsequently are connected to “mid” address (reason, why exactly this nodes, will be explained later in this chapter).

```
receiver1-> SetRecvCallback (MakeCallback (&ReceivePacket));
middle   -> SetRecvCallback (MakeCallback (&GrayHole));
```

SetRecvCallback function throw notification when new data is available to be read. As an attribute of the function is *Makecallback*. What is obvious, the goal is to allow one piece of code to call a function without any specific inter-module dependency. The function *ReceivePacket* and *GrayHole* are mentioned below.

```
for( int a = 1; a < 10; a = a + 1 )
{
        source -> Send(Create<Packet> (100));
        source2-> Send(Create<Packet> (100));
}
```

As the third step (after the allocation IP addresses and initiation the connection) is to send the packet with defined the size of payload as 100 (but it’s completely up to user). In the example, I am trying to send 10 packets from two different sources, however command *send* is unable to forward more than 3 packets from the same source node. The same result occurred even when I changed the duration of the whole simulation.

```
void GrayHole (Ptr<Socket> socket)
{
 Address from;
 Ptr<Packet> packet= Create <Packet> (100);
 socket->RecvFrom (from);
 InetSocketAddress senAddr = InetSocketAddress::ConvertFrom (from);
```

Firstly, *GrayHole* void function is called with defined local address “from”, which refers to IP address of received socket. We obtain it by calling the function *RecvFrom.* As a next step, this address is transformed to InetSocketAddress by function *ConvertFrom* with “from” address as an input.

```
 if (senAddr.GetIpv4() == "192.168.0.1") {
        std::cout<<"Paket s zdroj. adr. 192.168.0.1 bol zachyteny a zahodeny\n";
 }
 else  {
        std::cout<< "Paket s zdroj. adr.: " << senAddr.GetIpv4() <<
                        " bol preposlany\n";
        InetSocketAddress remote = InetSocketAddress ("192.168.0.6", 80);
        socket->Connect (remote);
        socket->Send(packet);
 }
}
```

As it is obvious from the code, if the sender address matches to IP “192.168.0.1”, the packet will be dropped as a result of gray-hole attack. If not, the packet will be forward to remote destination IP address “192.168.0.6” on the port 80.

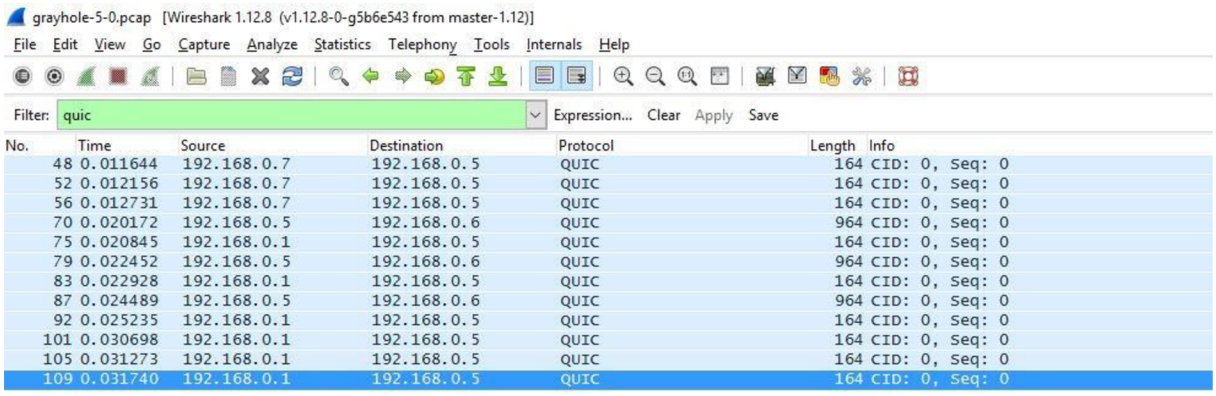
```
void ReceivePacket (Ptr<Socket> socket){
  Address from;
  socket->RecvFrom (from);
  InetSocketAddress senAddr = InetSocketAddress::ConvertFrom (from); //zdrojova adresa

  std::cout<<"Paket s zdroj. adr.: " << senAddr.GetIpv4() <<
                    " bol prijaty v cieli\n";
}
```

If the packet passes the test and its destination address is not equal “192.168.0.1”, packet will be forward to the node (in our case called “receiver”). I should mention the fact that gray-hole node works as a particular server node with own address and not as flown control mechanism. This causes all of the forwarded packets arrive to destination with source IP address of the malicious node-192.168.0.5 and not with address of accurate sender node. After running the program, output will look as follows:

```
Waf: Leaving directory `/home/denis/Desktop/ns-allinone-3.21/ns-3.21/build'
'build' finished successfully (5.063s)
Paket s zdroj. adr.: 192.168.0.7 bol preposlany
Paket s zdroj. adr.: 192.168.0.7 bol preposlany
Paket s zdroj. adr.: 192.168.0.7 bol preposlany
Paket s zdroj. adr.: 192.168.0.5 bol prijaty v cieli
Paket s zdroj. adr.: 192.168.0.5 bol prijaty v cieli
Paket s zdroj. adr.: 192.168.0.5 bol prijaty v cieli
Paket s zdroj. adr. 192.168.0.1 bol zachyteny a zahodeny
Paket s zdroj. adr. 192.168.0.1 bol zachyteny a zahodeny
Paket s zdroj. adr. 192.168.0.1 bol zachyteny a zahodeny
```

Pict. 18: Output after running gray-hole attack



Pict. 19: Captured PCAP file of node num. 5

Attached picture above shows the captured QUIC packets over the network, which represent packets manually created in the code. QUIC protocol (Quick UDP Internet Connections) provide multiplexed connections between 2 devices over UDP. It brings many benefits- security protection equivalent to TLS/SSL, reduced connection, transport latency and bandwidth estimation in each direction to avoid congestion. [29]

5.2.2 Mitigating the impact of the gray-hole

One of the technique proposed by Jaydeep Sen based on alarm and exploring alternative neighbour paths. If a node is marked as a malicious, a notification mechanism [Alarm Message] comes to play to send messages to all the nodes that are not yet suspected to be malicious. It results in isolation of malicious node and it is cut off from any network resources. [27] The mechanism consists of four security procedures which are invoked sequentially. Procedures consists of several steps:

- Neighbourhood data collection
- Local anomaly detection
- Cooperative anomaly detection
- Global alarm raiser

Malicious node stays isolated from the network by resending the alarm messages which can lead to overhead in the network.

Conclusion

In this thesis were shown samples of simple topology with the implementation of AODV routing protocol that was chosen with respect to the size of the network and also a visualization of the topology in the additional program NetAnim.

According to the obtained results, we can say that simulating the black-hole attack via setting the malicious node down is not properly way. Because the black-hole should behave as a normal node for its neighbours and blindly drop the packets after obtaining. This option was chosen by reason of impossibility of implementing the command *SetForwarding.* If we don't take into account the routing tables, where some nodes occurred with the flag "down", we can observe the same result- dropping all of the incoming packets. Also, it was proposed a solution for eliminating the influence of the black-hole.

As regards the gray-hole, my focus was aimed to develop appropriate model of GH in source code. I have chosen the variant „node dependency" attack where packets are filtering by IP address or port. Since I did not find functional method how to implement flow control of traffic generated by applications, I was forced to look at the packets as objects, generate them manually and check a match in the middle node. As a result, output doesn't refer to original approach of GH. Anyway, take this thesis as my own implementation of GH.

References

[1] Using the Logging Module: Logging Overview [online], 2015 [cit. 2015-10-18]. Available from: https://www.nsnam.org/docs/release/3.7/tutorial/tutorial_21.html

[2] ns3::GridPositionAllocator Class Reference [online], 2015 [cit. 2015-10-28]. Available from: https://www.nsnam.org/doxygen/classns3_1_1_grid_position_allocator.html

[3] Ns-3 Tutorial, release ns-3.18.1 [online], 2013 [cit. 2015-11-4]. Available from: https://www.nsnam.org/docs/release/3.18/tutorial/ns-3-tutorial.pdf

[4] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, Jang-Ping Sheu, The Broadcast Storm Problem in a Mobile Ad Hoc Network [online]. Available from: http://www.cs.berkeley.edu/~culler/cs294-f03/papers/bcast-storm.pdf

[5] ns-3 Manual, Release ns-3-dev [online], 2013 [cit. 2015-11-10]. Available from: https://www.nsnam.org/docs/manual/ns-3-manual.pdf

[6] VADKERTI, G. VYTVOŘENÍ EXPERIMENTÁLNÍ MANET SÍTĚ V SIMULÁTORU NS-3. Brno, 2013. Bachelor's thesis. FEKT VUT Brno.

[7] Manaseer, Saher, On backoff mechanisms for wireless Mobile Ad Hoc Networks, 2010 PhD thesis, University of Glasgow.

[8] BHATTACHARYYA, A., BANERJEE, A., BOSE, D., SAHA, H., N., BHATTACHARJEE, D. Different types of attacks in Mobile ADHOC Network: Prevention and mitigation techniques. https://ipfs.io/ipfs/QmfXH9XtP7xmoTH8WAp4HNSduqWMwLTH8B8TvbTkdgzNAa/cc-by-nc-sa-3.0/1111/1111.4090.pdf

[9] Fan-Hsun Tseng, Li-Der Chou1, Han-Chieh Chao, A survey of black hole attacks in wireless mobile ad hoc networks [online], 2011, [cit. 2015-11-13]. Available from: http://www.hcis-journal.com/content/pdf/2192-1962-1-4.pdf

[10] Shani Makwana, Krunal Vaghela, Cooperative Gray Hole Attack Detection and Prevention Techniques in MANET: Review [online], 2013, [cit. 2015-11-13]. Available from: http://www.ijsr.net/archive/v4i1/SUB15152.pdf

[11] Magnus Frodigh, Per Johansson, Peter Larsson, Wireless ad hoc networking—The art of networking without a network [online], 2000, [cit. 2015-11-13]. Available from: http://www.ericsson.com/ericsson/corpinfo/publications/review/2000_04/files/2000046.pdf

[12] Routing. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-11-17]. Available from: https://en.wikipedia.org/wiki/Routing

[13] Basu Dev Shivahare,Charu Wahi , Shalini Shivhare, Comparison Of Proactive And Reactive Routing Protocols In Mobile Adhoc Network Using Routing Protocol Property [online], 2012, [cit. 2015-11-13]. Available from: http://www.ijetae.com/files/Volume2Issue3/IJETAE_0312_59.pdf

[14] Harjeet Kaur, Varsha Sahni, Dr. Manju Bala, A Survey of Reactive, Proactive and Hybrid Routing Protocols in MANET: A Review [online], 2013, [cit. 2015-11-16]. Available from: http://www.ijcsit.com/docs/Volume%204/vol4Issue3/ijcsit2013040326.pdf

[15] Ad hoc On-Demand Distance Vector Routing. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-11-22]. Available from: https://en.wikipedia.org/wiki/Ad_hoc_On-Demand_Distance_Vector_Routing

[16] Reactive protocols – AODV [online], 2004, [cit. 2015-11-22]. Available from: http://www.olsr.org/docs/report_html/node16.html

[17] Optimized Link State Routing Protocol In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-11-23]. Available from: https://en.wikipedia.org/wiki/Optimized_Link_State_Routing_Protocol

[18] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, L. Viennot, Optimized Link state Routing Protocol for Ad Hoc Networks [online], [cit. 2015-12-3]. Available from: http://www.cs.jhu.edu/~dholmer/600.647/papers/OLSR.pdf

[19] Nicklas Beijar, Zone Routing Protocol (ZRP), [online], 2015, [cit. 2015-12-3]. Available from: http://www.netlab.tkk.fi/opetus/s38030/k02/Papers/08-Nicklas.pdf

[20] Virendra P. S. , Sweta J., Jyoti S., Hello Flood Attack and its Countermeasures in Wireless Sensor Networks [online], 2010, [cit 2016-02-8]. Available from: http://ijcsi.org/papers/7-3-11-23-27.pdf

[21] C.Dhivya Devi , B.Santhi, Study on Security Protocols in Wireless Sensor Networks, [online], 2013, [cit 2016-02-10]. Available from: http://www.enggjournals.com/ijet/docs/IJET13-05-01-097.pdf

[22] A.Shrivastava, R. Dubey, Wormhole Attack in Mobile Ad-hoc Network: A Survey, [online], 2013, [cit 2016-02-17]. Available from: http://www.sersc.org/journals/IJSIA/vol9_no7_2015/26.pdf

[23] D. Dong, M. Li, Y. Liu, X. Li, X. Liao, Topological Detection on Wormholes in Wireless Ad Hoc and Sensor Networks, [online], [cit 2016-02-29]. Available from: http://www.cse.ust.hk/~liu/TON-wormhole.pdf

[24] S. Shrivastava, Rushing Attack and its Prevention Techniques, [online], 2013, [cit 2016-02-29]. Available from: http://www.ijaiem.org/Volume2Issue4/IJAIEM-2013-04-27-082.pdf

[25] Sockets APIs, Ns-3 vns-3.11 documentation, 2011,[cit 2016-02-29]. Available from: https://www.nsnam.org/docs/release/3.11/models/html/sockets-api.html

[26] Ns3::Socket Class Reference, A Discrete-Event Network Simulator ns-3-dev, 2016, [cit 2016-02-29]. Available from: https://www.nsnam.org/doxygen/classns3_1_1_socket.html#a680c1edd8b3ec3adf4085e1ded1e8a0d

[27] N. Patidar, P. Jain, An Approach To Mitigate Gray-hole Attack In Mobile Ad-hoc Networks, [online], 2016, [2016-03-29]. Available from: http://ijseas.com/volume2/v2i3/ijseas20160307.pdf

[28] Pcap: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-04-01]. Available from: https://en.wikipedia.org/wiki/Pcap

[29] Quic: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-04-01]. Available from: https://en.wikipedia.org/wiki/QUIC

List of used abbreviations

- AODV Ad-hoc on demand distance vector
- CTS Clear-to-send
- RTS Request-to-send
- Manet Mobile ad-hoc network
- MPR Multipoint relays
- OLSR Optimized link-state routing
- ZRP Zone Routing Protocol
- QoS Quality of Service
- RERR Route error
- RREP Route reply
- RREQ Route request
- TCP Transimission control protocol
- BRP Broadcast Resolution Protocol
- IERP Inter Zone Routing Protocol
- MAC Media access control
- XML eXtensible Markup Language
- PCAP Packet capture

List of appendixes