AALBORG UNIVERSITY
DENMARK

AALBORG UNIVERSITY

Innovative Communication Technologies and Entrepreneurship

(ICTE)

DDoS Attack Detection in SDN-based VANET Architectures

Martina Stoyanova Todorova and Stamelina Tomova Todorova

Supervisor: Professor Albena Mihovska

Co-supervisor: Research Assistant Anton Katov

June 2016

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our supervisor Professor Albena Mihovska and our co-supervisor Researcher Assistant Anton Katov of Center for TeleInFrastructures (CTIF) faculty at Aalborg University. Due to their continuous support during our research, positive attitude, motivation, enthusiasm, and immense knowledge, without which the validation survey could not have been successfully conducted.

We would also like to thank Professor Ramjee Prasad and Professor Vladimir Pulkov for the great opportunity to be part of the collaboration program between Aalborg University and Technical University of Sofia.

Finally, we must express our very profound gratitude to our families for providing us with unfailing support and continuous encouragement throughout our years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you.

ABSTRACT

Software-defined networking (SDN) is an emerging technology, which provides network architecture that decouples the control plane from the data plane. This main characteristic of SDN are bringing several of advantages. Due to the centralized control the network becomes more dynamic, and the network resources are managed in more efficient and cost-effective manner.

Another technology that focuses the attention of both the industry and the academy and has a huge potential to be wildly used all over the world is Vehicular Ad Hoc Networks (VANET). It is based on Mobile Ad Hoc Networks (MANET), in which the nodes are considered to be vehicle instead of mobile devices. VANETs are the key components of the intelligent transport systems (ITSs), whose major aim is to improve road safety and to provide different applications to the drivers and the passengers.

One of the main objectives of this thesis is to investigate how these two technologies can be implemented together, in order to achieve improved network performance. We claim that VANET networks can benefit from using SDN controller. Due to the separation between the control and data planes in VANET, network intelligence can be logically centralized and the underlying network infrastructure can be decoupled from the applications.

The centralized control of SDN brings an immense number of advantages, but it also can become a single point of failure of the network. The entire network could be compromised if the controller is under attack and therefore the network security in SDN-based VANETs is a major concern. In order to address some major security aspects of the VANET scenario, we estimate how Denial of Service Attack (DoS) and the Distributed Denial of Service Attack (DDoS) can influence the performance of SDN-based VANET network. The main purpose of this work is to detect DDoS attack of User Datagram Protocol (UDP) packets in order to meet the needs of real-time services, such as accident prevention, traffic jam warning, or communication.

This diploma thesis designs and tests a DDoS detection algorithm for SDN-based VANET networks. The test scenarios include launching normal and DDoS attack traffic with spoofed source IP addresses. Based on traffic features, entropy is used to measure the degree of randomness of occurrence of destination IP address of the packets. The algorithm is implemented as a software

module on the SDN controller, by the means of two additional functions for detection of DDoS attacks. Entropy is calculated within predefined window size to measure uncertainty in the coming packets. After that the result is compared to a predefined threshold in order to classify the traffic as normal or attack traffic.

.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AAA - authorization, authentication, and accounting

ACK – acknowledgment

ALTO - Application Layer Traffic Optimization

AP - Access Point

API - Application Programming Interface

ARME - Apache remote memory exhaustion

ARP - Address Resolution Protocol

BGP - Border Gateway Protocol

BSS - Basic Service Set

BSSID - Basic Service Set Identification

C&C - Command and Control Channels

CCH - Control channel

DA - Destination Address

DDoS - Distributed Denial-of-Service

DNS - Domain Name System

DoS - Denial-of-service

DPI - Deep Packet Inspection

DrDoS - Distributed Reflection Denial of Service

DS - Distribution Service

DSRC - Dedicated Short-range Communications

ECC - Elliptic Curve Cryptography

EDGE - Enhanced Data Rates for GSM Evolution

ELP - Electronic License Plate

ESS - Extended Service Set

EVDO - Evolution Data Optimized or Enhanced Voice Data Only

GPRS - General Packet Radio Service

HSDPA - High Speed Downlink Packet Access

IaaS - Infrastructure as a Service

ICMP - Internet Control Message Protocol

IDSs - Intrusion Detection Systems	
IEEE - Institute of Electrical and Electronics Engineers	
IP – Internet Protocol	
IPSs - Intrusion Prevention Systems	
ITS - Intelligent Transportation Systems	
LISP - Locator/Identifier Separation Protocol	
LLC - Logical Link Control	
LTE - Long Term Evolution	
MAC - Media Access Control	
MANETs - Mobile Ad Hoc Networks	
MPLS - Multiprotocol Label Switching	
MTD - Moving Target Defence	
NAT - Network Address Translators	
NIDS - Network Intrusion Detection System	
NOS - Network Operating System	
NTP - Network Time Protocol	
NVP - Network Virtualization Platform	
OBU - On Board Unit	
OF - Open Flow	
OFDM - Orthogonal Frequency Division Multiplexing	
ONF - Open Networking Foundation	
OS - Operating System	
OSI - Open Systems Interconnection	
PDA - Personal Digital Assistants	
PDU - Packet Data Unit	
PHY - Physical Layer	
PLCP - Physical Layer Convergence Protocol	
PMD - Physical Medium Access	
PPDU - Protocol Packet Data Unit	
PPS - Packets Per Second	
QNOX - QoS-aware Network Operating System	

QoS - Quality of Service

RA - Receiving STA Address

RADIUS - Remote Authentication Dial In User Service

RAM - Random Access Memory

REST - Representational state transfer

RFID - Radio-Frequency Identification

RPS - requests per second

RSU - Road Side Units

SA - Source Address

SDN - Software-Defined Networking

SOM - Self-Organizing Maps

SSID - Service Set Identification

SYN - Synchronization

TA - Transmitting STA Address

TCAM - Ternary Content Addressable Memory

TCP - Transmission Control Protocol

TLS - Transport Layer Security

UDP - User Datagram Protocol

UMTS - Universal Mobile Telecommunication System

V2V - Vehicle-to-vehicle

VANETs -Vehicular Ad Hoc Networks

vCRIB - virtual Cloud Rule Information Base

VLANs - Virtual Local Area Networks

VM – Virtual Machine

VoIP – Voice over Internet Protocol

WAVE - Wireless Access in Vehicular Environments

WLANs - Wireless Local Area Networks

XSP - eXtensible Session Protocol

1 INTRODUCTION

1.1 INTRODUCTION

The fast development of technologies nowadays requires the development and implementation of new networking strategies. After examination of the new networking trends, we found that SDN is an emerging technology, and its uniqueness comes by the fact that it provides programmability through decoupling of control and data planes, and ensures simple programmable network devices, instead of making them more complex. SDN offers a new convenient platform for implementing and testing new ideas and encourages innovative network design, using its network programmability and the fact that isolated virtual networks can be defined by the control plane. It allows a real-time centralized control due to its ability to receive current network status.

VANET is another rapidly growing technology that has gained a lot of popularity among the industry and academic research community. It is based on MANET, in which nodes are vehicular instead of mobile devices. VANETs are key component of ITS and is seems to be the most valuable concept for improving efficiency and safety for future transportations. Its exclusive characteristics include predictable mobility, no power limitations, variable network density, high computational ability, rapid changes in network topology and etc.

After a thorough and comprehensive analysis of this two networking trends we decided to lead our efforts into SDN-based VANET direction. Despite all the above mentioned benefits of these two technologies, and the subsequent implementation of both for better performance, we believe that security is one of the most challenging concern. The centralized control of SDN can be considered as one of the main benefits, but the entire network could be compromised if the controller is under attack. For that reason we decided to launch a DDoS attack on the control layer in SDN-based VANET network. Nevertheless, there are many threat vectors in both technologies, but one common is represented by DoS and DDoS attacks.

In the literature a lot of DDoS attack detection methods based on different traffic features are proposed, and in this thesis we examine some of them. The idea of using entropy to measure the randomness in destination IP addresses took our attention and we focused our efforts on implementing a DDoS detection algorithm for SDN-based VANET scenario. The algorithm is implemented into the controller, by means of additional functions for detecting the attack traffic.

The experiments are performed and the obtained results show the effectiveness of proposal DDoS attack detection solution.

1.2 MOTIVATION

In the beginning of our research the focus was directed at examining and exploiting new and fast development technologies. After deep research and analysis of networking trends we decided to lead our efforts into SDN based VANET direction. These are two emerging technologies, which are still under consideration and development. For us it was interesting to explore the potential combination between them, since the research on that field are in its early stage.

Nowadays, the network structure itself allowed for many security trends to occur. Therefore the network security can be consider as a major concern. For that reason we decided to estimate how the existing attack methods can influence the performance of SDN based VANET network.

1.3 PROBLEM STATEMENT

At present, network security can be considered a major issue in every networking application. For this reason, when we take into consideration the SDN-based VANET scenario, we focused our efforts on identifying the major security concerns related to the SDN and VANET technologies, both separately and together. After deep analysis of the security challenges related to these two technologies, we chose to explore the significance of DDoS attacks on the control layer. There are many threat vectors that can affect them, but one common is represented by DoS and DDoS attacks. This type of attacks can seriously influence the performance of SDN-based VANET network, and due to the centralized control the entire network could be compromised if the controller becomes unreachable.

Due to the aforementioned reasons this thesis designs and tests a DDoS detection algorithm for SDN-based VANET network. The tests scenario includes launching a DDoS attack and implementing a DDoS detection module on the SDN controller. Different methods are already proposed in the literature for DDoS detection, but for the purpose of our work we decide to use entropy, and the main reason for that is its ability to measure randomness in a network.

In order to implement this algorithm with entropy, a thorough analysis of VANET technology is extremely important, to establish an SDN topology that is applicable in realistic scenarios. Furthermore, the detection parameters will be selected, so as to satisfy the requirements of real-time services of VANET, such as accident prevention, traffic jam warning, or communication.

1.4 SCOPE

This thesis is organized as follows:

Chapter 2 includes an overview of SDN basic concept, definitions and the benefits of its usage and implementation. Later, SDN architecture model discussed. Last part of this chapter, each layer of SDN architecture separately are covered.

Chapter 3 contains thorough review of VANETs definition, standard, and applications. For the purpose of our work a survey on Software-Defined VANET network and Software-Defined VANET applications are also presented.

Chapter 4 describes the security challenges in SDN and VANET, followed by deep analysis of DoS and DDoS attack and also a new trend of launching those type of attacks, namely “botnets”. Finally, different types of DDoS attacks in SDN networks are examined.

In Chapter 5, in depth analysis of different types of anomaly detection has been done. An introduction of the entropy as a term and how it is used in several existing methods for DDoS detection, as well as its benefits for measuring the randomness are included there. Based on that, in this chapter is presented our proposed topology for DDoS attack detection in SDN - based VANET scenario.

Chapter 6 shows the procedure of creating the experiment, the results from the simulation and evaluation of the proposed algorithm for DDoS detection in SDN VANET scenario.

Chapter 7 presents conclusion as a summary of the contribution to this thesis, and discusses the future prospects and open problems which can be further investigated and explored.

1.5 RESEARCH CONTRIBUTION

In this thesis, we presented SDN-based VANET architecture design and we demonstrated the influence of a DDoS attack in such scenarios. The main contribution of our work is the implementation of DDoS detection mechanism based on measuring the entropy of destination IP address on SDN controller in several test cases. After deep research on the existing methods for DDoS attack detection, we conclude that entropy is a convenient approach. The entropy is measured based on traffic features and we believe that in SDN environment, where the controller is able to obtain instant information by the network, this method can be easily implemented and developed. Furthermore, we tested the proposed solution to estimate if it gives reliable results in detection of DDoS attack launched via botnet. We simulated various experiments, as our goal was to obtain results as close as possible to a real world scenario. The final results show that the proposed method is adequate and effective for DDoS detection in VANET scenario.

Based on our tests, we believe that in dynamic environment, like VANET networks, this algorithm against DDoS attacks can be used as a basis for the development of real-time detection mechanisms, which can update the parameters instantly, after obtaining current network status information.

The results of this thesis are currently being submitted as a research paper to the GWS 2016 to be held in Dubrovnik, Croatia, on Sept 18-21, 2016.

2 SOFTWARE-DEFINED NETWORKING

2.1 DEFINITION AND BENEFITS OF SOFTWARE-DEFINED NETWORKING

2.1.1 DEFINITION OF SDN CONCEPT

The Open Networking Foundation (ONF) [1] is a non-profit consortium that is devoted to development, standardization, and commercialization of Software-defined Networks (SDN). ONF gives the most clear and well received definition of SDN: SDN is an emerging network architecture where network control is decoupled from forwarding and is directly programmable [2]. According to this definition, SDN can be defined by two characteristics: decoupling of control plane from the data plane and programmability on the control plane, but neither of these two signatures of SDN is fully new in network architecture, as detailed in the following.

In [3] the authors had made a comprehensive survey of these efforts. According to their paper, it can be said that several previous work have been made into network programmability. For instance SwitchWare [4], [5] uses the concept of active networking solution and that allows packets transmitting through the network to dynamically modify its operations. Similar to this work, Click [6], XORP [7], Quagga [8], and BIRD [9], also attempt to build extensible software routers by making network devices programmable. By using different or modifying existing routing software the behaviour of these network devices can be controlled.

Furthermore, the idea of decoupling control from data planes has been explored during the last decade. In [10] the concept of Border Gateway Protocol (BGP) inter-domain routing is replaced by centralized routing control. In this way the authors claim that the complexity of fully distributed path computation are reduced. At that time, IETF proposes the Forwarding and Control Element Separation (ForCES) framework. They separates control and packet forwarding elements [11]. A 4D approach had been proposed in [12]. This method consists a concept of network architecture with four planes. These planes are called “decision”, “dissemination”, “discovery”, and “data”. In 2007, Ethane work were presented [13], it relies on using simple flow-based Ethernet switches supplemented with a centralized controller. In this way the routing flows can be managed.

The uniqueness of SDN concept comes by the fact that it provides programmability through decoupling of control and data planes - SDN provides simple programmable network devices instead of making networking devices more complex. In addition to that, SDN offers separation of control plane and data plane in the network architecture. With this model, the control of the network can be done separately on the control plane without impact on the data flows. The intelligence of the network can be removed from the switching devices and placed on the controller. Meanwhile, the switches can be controlled externally by software without need of on-board intelligence. The separation of control from data planes provides not only a simpler programmable environment but a greater freedom for external software to define the behaviour of a network as well.

2.1.2 BENEFITS OF SDN

SDN, with its intrinsic separating of control from data planes, brings a greater control of a network using programming.

This feature would lead to potential benefits of increased configuration, improved performance of the network, and also works in favour of innovation in network design and operations [3]. These benefits are summarized in Table 2.1.

	SDN	**Conventional Networks**
Features	- separated data and control plane - programmability	- a new protocol for every problem - complex network control [14]
Configuration	automated configuration with centralized validation	error manual configuration
Performance	dynamic global control with cross layer information	- limited information -relatively static configuration
Innovation	- easy implementation of the software for innovations - sufficient test environment with isolation - rapid deployment using upgrade of the software	- hard hardware implementation for innovation - limited testing environment - long standardization process

Table 2.1 Benefits of SDN

SDN can break the layering barrier because the control includes also link tuning at a data link level, in addition to packet forwarding at a switching level. Furthermore, it allows a real-time centralized control of the network due to its ability to receive current network status. This control is based also on a user defined policies. On the other hand this brings improvements in network configuration and network performance. SDN provides a new convenient platform for implementing and testing new ideas and encourages innovative network design using its network programmability and the fact that isolated virtual networks can be defined by the control plane. Next, these above mentioned benefits of SDN are examined in details.

Enhancing Configuration - The configuration can be seen as one of the most important functions of the network management. Adequate configuration is required to achieve good network operation, when some new equipment is added to the network. Due to the heterogeneity among manufacturers of network devices and interfaces for configuration, current network configuration usually requires a manual configuration procedure, which is tough and error prone. Meanwhile, to handle this configuration errors a serious effort to troubleshoot has to be done. Automatic and dynamic reconfiguration of a network is still a considerable challenge with the conventional network design and SDN can be a solution to this situation in network management. In SDN the configuration of network devices can be done automatically via software controlling from a single point. This is possible because of the unification of the control plane over all kinds of network devices such as switches, routers, Network Address Translators (NATs), firewalls, and load balancers [15]. So, based on the current network status the entire network can be programmatically configured and dynamically optimized.

Improving Performance - From the perspective of network operations one of the main objectives is to maximize performance of the present network infrastructure. This is not an easy job, because of coexistence of various technologies and stakeholders in a single network. The current effort is focused on improving performance of a subset of networks or the quality of user experience for some network services. It can be said that these methods are not fully satisfying, because they are based on local information without cross-layer consideration. The appearance of SDN can provide the opportunity to optimize globally the network performance, due to the centralized control with a global network view. In SDN a feedback control also exists. It works with information exchanged between different layers in the network architecture. These features of SDN leads to proper designed centralized solutions for the many challenging performance optimization problems. Well known, classical problems such as data traffic scheduling, end-to-end congestion control, load balanced packet routing, energy efficient operation, and Quality of

Service (QoS) support, can now be easily solved and deployed to verify their effectiveness in improving network performance.

Encouraging Innovation - Nowadays we can notice the continuing evolution of network applications. Therefore the future networks have to encourage innovation instead of attempting to make a precise predictions to meet perfectly the requirements of future applications [16]. In current networks, when a new idea comes up immediately faces challenges in implementation, experimentation, and deployment. The general obstacle emerges from widely used proprietary hardware in existing network components and that prevents modification for experimentation. Further, even in case when these experimentations are possible, they are usually conducted in a separate simplified testing environment. These testing does not provide acceptable confidence for industrial implementation of these innovations or network designs. There are some efforts to enabled large scale experimentations, but they cannot completely handle the problem.

In contrast, SDN provides a programmable network platform to encourage innovation. The implementation [17], experimentation [18], deployment of new ideas and applications can take advantage of this function of SDN network. The experiments on a real environment is achievable due to the high configurability of SDN, because this it provides good separation among virtual networks. The transition from an experimental to an operational phase of the deployment of innovations may be performed smoothly.

2.2 SDN ARCHITECTURE MODEL

ONF has suggested a reference model for SDN networks [1]. It is illustrated on Figure 2.1. The architecture consists of three layers, stacking over each other:

- Infrastructure layer
- Control layer
- Application layer

One of the main differences between SDN and traditional networks is that the physical devices now are just forwarding elements without control functions or any software. The intelligence of the network is removed from the data plane devices to the control plane. Moreover, this networks are built on a top of open standard interfaces (Open flow and e.g), which is a crucial for enabling the communication and configuration between data plane and control plane devices. These open interfaces allow control plane to dynamically configured heterogeneous forwarding devices, which is very difficult in traditional networks. There are two main elements in SDN architecture: the controllers and the forwarding devices. The former is a software stack, while a data plane device is a hardware of software element, which are specialized in packet forwarding.

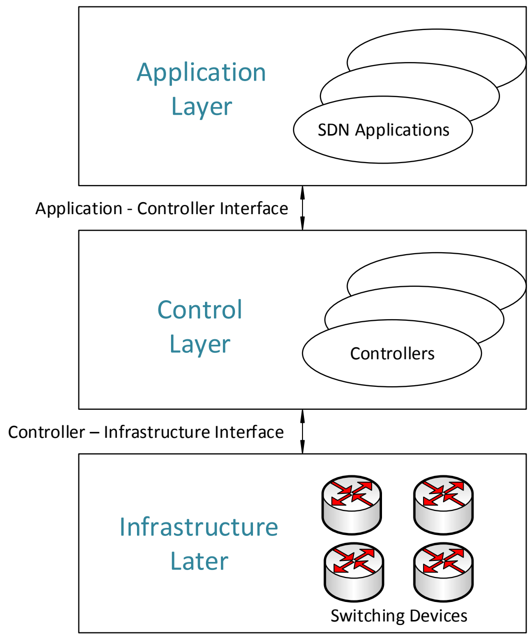


Figure 2.1. SDN Reference Model

The infrastructure layer is composed of switching devices such as switches, routers, etc. in the data plane. In the general case these switching devices have two functions. The first function is collecting network status, storing it temporally and after that sending the information to controllers. These status data can include information about network topology, traffic statistics, and usages of the network. The second function is responsible for processing packets. The packets are treated based on flow rules that are provided by the controller.

The control layer is connected to the application and the infrastructure layers, via two interfaces. For interacting with the infrastructure layer it uses the south-bound interface. It specifies features for controllers to connect switching devices functions. These features usually consist reporting network status and importing forwarding rules to process the packets. For upward interacting with the application layer the control plane uses the north-bound interface. It provides service access points in different forms, for instance, an Application Programming Interface (API). SDN applications can access data for network status, which is reported from infrastructure layer devices through this API, to make system adjustment decisions based on it. By setting packet forwarding rules to switching devices using this API these decisions can be executed. In case of a large administrative network domain multiple controllers will exist and they are connected via east-west interface. It is required because the controllers have to share network information and coordinate their decision-making processes [19].

To meet the requirements of the users applications are designed and placed on the application layer. These applications can access and control the infrastructure layer devices using the programmable platform of the control layer. For instance, SDN applications may include dynamic access control, seamless mobility and migration, server load balancing, and network virtualization.

In the following subsections a detailed description of each SDN layer has been made in order to acquire a deep understanding of the SDN architecture model.

2.3 DATA PLANE

For the purpose of our work a deep understanding of OpenFlow standard is required, because the simulations are based on it.

Open Flow (OF) is an open communications protocol which provides access to the data plane in SDN architecture. Often when we talk about SDN we mean Open Flow, but also we have to be aware of that SDN can work with other mechanisms. SDN consists of decoupling the control from data plane, whereas OpenFlow describes how software controller and a switch should communicate in an SDN architecture. Open Flow enables controller to define the path of packets through the network of switches. Furthermore, Open Flow can use switches from different suppliers to be managed remotely using a single, open protocol. A main capability of OpenFlow networks is the possibility of dynamic updates of forwarding rules. Different types of changes can be performed in real time. The decision for these changes are taken by a software controller.

This section of our project provides information about the components and some functions of the Open Flow switch and how it can be managed by the remote controller.

Figure 2.2 shows the main components of the Open Flow switch and how it interact with the controller. Open Flow switch consists minimum one Flow Table, also a Group Table and an Open Flow Channel to the controller [20]. The communication between the switch and the remote controller is carried out by Open Flow Protocol. Via this open interface the controller can manage the behaviour of the switches. This management includes adding, deleting and updating flow entries in flow tables. Also through this channel, the controller receives packets from the switches and sends packets to the switches.

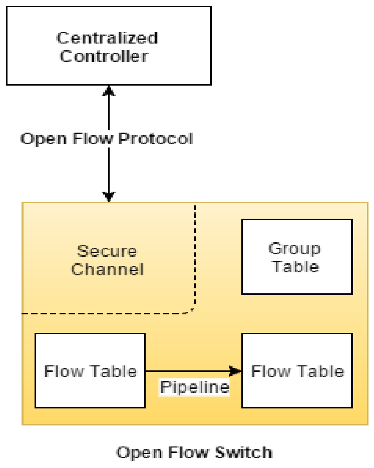


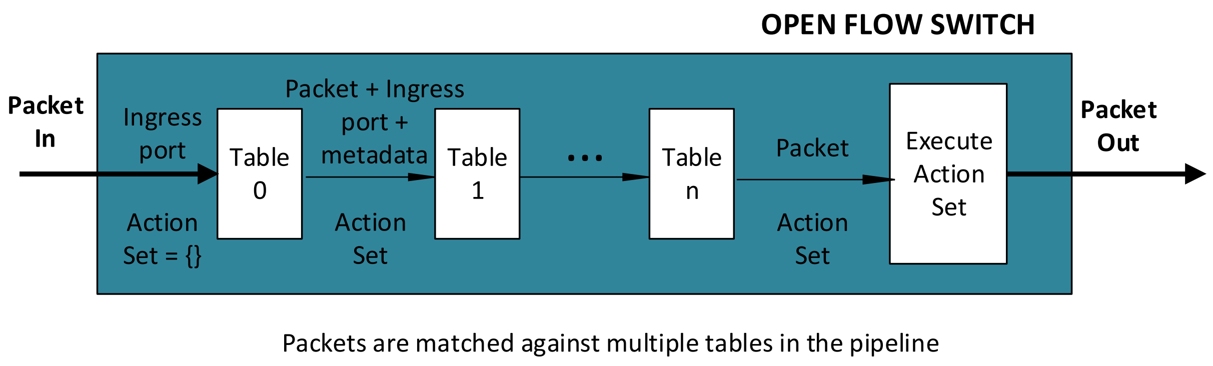
Figure 2.2 Main components of an OpenFlow switch

Open Flow networks treat all traffic as a flows. Switches use flow tables to forward the packets. A flow table is a list of flow entries. The entries contain match fields, counters and set of instructions. Figure 2.3 shows the main components of a flow entry. Match Fields includes the ingress port and packet headers. Also, sometimes it consist of metadata specified by a previous table. Incoming packets are compared with the match fields. Counters updates when packets are matched. The instructions in the flow entry modify the action set or pipeline processing. A flow entry includes a priority part which matching precedence of the current entry. Timeouts shows the maximum time before flow is expired by the switch. The last part in the flow entry is the cookie part, which may be used by the controller to filter, modification or flow deletion. This opaque data is not used when processing packets. Further in this chapter the components will be explained in-depth.

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

Figure 2.3 Main components of a flow entry in a flow table

There are two types of switches - OpenFlow–only and OpenFlow-hybrid. The first supports only OpenFlow operations, while the second supports both OpenFlow operations and normal Ethernet switching operations. A switch uses Ternary Content Addressable Memory (TCAM) and Random Access Memory (RAM) to process each packet. Every switch may contains multiple flow tables (every flow table contains multiple flow entries). This sequence of flow tables are called OpenFlow Pipeline. Flow tables are numbered, starting at 0 and the pipeline processing begins at the first flow table. (Figure 2.4)



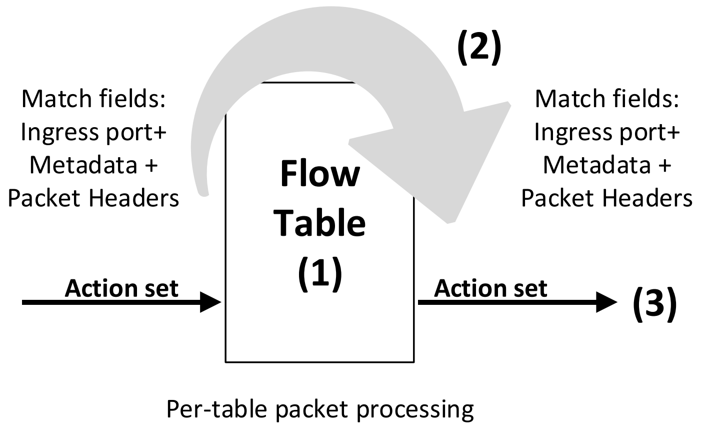

(2)
Match fields: Ingress port+ Metadata + Packet Headers
Flow Table (1)
Match fields: Ingress port+ Metadata + Packet Headers
Action set
Action set
(3)
Per-table packet processing

Figure 2.4 Packet flow through the processing pipeline

(1) Find highest-priority matching flow entry

(2)Apply instructions

- Modify packet & update match fields
- Update action set
- Update metadata

(3) Send match data and action set to next table

If the packet matched against the flow entry when processed by a flow table, the instructions set included in that entry is executed. This instructions can lead the packet to another flow table, where different instructions can be performed. A flow entry can only direct the packet to a flow table with higher number than its own. It means that the pipeline processing can go only forward. If the entry does not lead the packet forward to another flow table, the pipeline processing stops and the action set associated with this packet is executed. This action set is empty by default, but the flow entry modify it using Write or Clear-Action instructions. Also this action set is carried between flow tables and when pipeline processing stops the actions are executed. There is a predefined order, which specify the order that the actions in the set are applied. This order is illustrated in Table 2.2. It is obvious that the output action is executed last, but if a group action is defined, the output action is ignored. If no output or no group action are specified, the packet will be dropped.

No.	Action
1	copy TTL inwards
2	pop
3	push-MPLS
4	push-PBB
5	push-VLAN
6	copy TTL outwards
7	decrement TTL
8	set
9	QoS
10	group
11	output

Table 2.2 Apply-Actions instruction

The flowchart on Figure 2.5 shows packet flow through an OpenFlow switch. According to this figure it can be said that when the data enter, the switch starts by performing a table lookup in the first flow table (table 0). Depending on pipeline processing, the switch can perform lookups in other flow tables. Packet match fields are extracted from the whole data and they are used for table lookups. This fields include several header fields. Also a metadata can be included in this match fields and it is used to carry information between the flow tables. If the packet is matched against the table, only the highest priority entry must be selected. After that the counter must be updated and the instructions must be applied. How the switch handle with a table miss (a packet does not match an entry) depends on the table configuration. A table-miss flow entry defines how to process unmatched packets. There are several options – dropping packets (using the Clear-Action instructions), passing them to another table or sending them to the controller (using the controller reserved port-the control channel). It is important to emphasize the latest version of Open Flow specification does not define the switch behaviour when a malformed or corrupted packet are received.

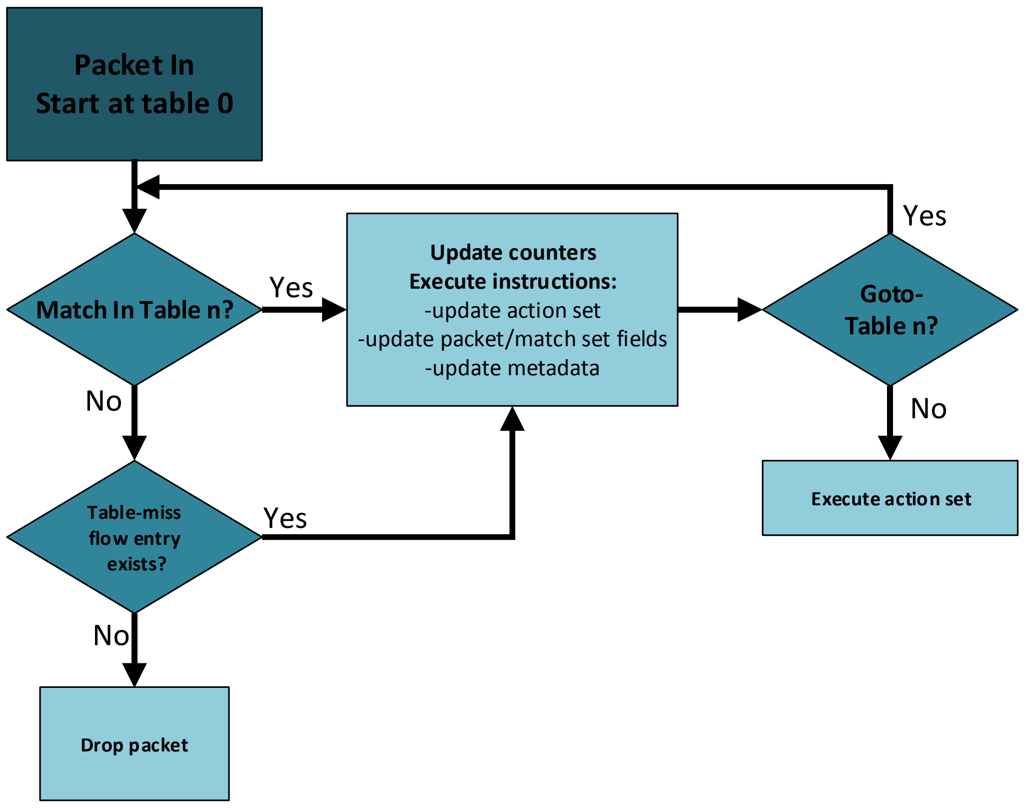


Figure 2.5 Flowchart detailing packet flow through an OpenFlow switch.

Flow entries can be removed only in three cases. First is when the controller requests that, the second way rely on switch flow expiry mechanism and the last one is the optional switch eviction mechanism. The controller can actively remove entries using the delete flow table modification massage or by removing a group. The switch flow expiry mechanism is based on the state and configuration of the flow entries. This mechanism is run by the switch. This way uses two types of field associated with the entry – hard_timeout and Idle_timeout to estimate if the flow entries have to be removed. The last way to remove a flow entries is when the switch needs to reclaim resources. Flow entry eviction is optional and works only on flow tables where it is enabled. No matter which mechanism is used to remove the entry, the switch must check the flow entry`s OFPFF_SEND_FLOW_REM flag and if it is set, a message is sent to controller. This removal massage contains detailed information about the flow entry, removal, duration etc,

On Figure 2.2 is shown that a group table can be used in OpenFlow switch. This type of table contains a group entries. On the following Table 2.3 presents the main components of a group entry.

Group Identifier	Group Type	Counters	Action Buckets
32 bit – integer uniquely identifying	To determine group semantics	Updated when packets are processed by a group	An ordered list of action buckets. Each of them contains a set of actions and associated parameters.

Table 2.3 Main components of a group entry in the group table

The switch is required to support only group types marked as "Required". They have two subtypes – all and indirect. "All" execute all buckets in the group, while the "Indirect" execute the one defined bucket in this group.

OpenFlow specification defines different counters for each flow table, flow entry, port, queue, group, group bucket, meter and meter band. They are summarized in the next table – Table 2.4 (a switching device is required to support only those counter, marked as "Required", therefore we include only them in the table):

Counter	Bits
Per Flow Table	
Reference Count (active entries)	32
Per Flow Entry	
Duration (seconds)	32
Per Port	
Received Packets	64
Transmitted Packets	64
Duration (seconds)	32
Per Queue	
Transmit Packets	64
Duration (seconds)	32
Per Group	
Duration (seconds)	32
Per Meter	
Duration (seconds)	32

Table 2.4 List of required counters in OpenFlow specification

As mentioned previously, when a packet matches the entry, for each flow are executed a set of instructions and these instructions lead to changes in the packet, action set and pipeline processing. In the next table (Table 2.5) are described all instruction types, but the switch has to support only those marked as "Required".

Instruction	Description	Optional/Required
Meter ***meter_id***	Direct packet to the specified meter (the packet can be dropped, depending on meter configuration and state)	Optional Instruction
Apply-Actions ***action(s)***	Applies the specific action immediately (without change to the Action Set)	Optional Instruction
Clear-Actions	Clears all the actions in the action set immediately.	Optional Instruction
Write-Actions ***action(s)***	Merges the specified actions into the current action set - overwrite or add it.	Required Instruction
Write-Metadata ***metadata / mask***	Writes the masked metadata value into the metadata field	Optional Instruction
Goto-Table ***next-table-id***	Indicates the next table in the processing pipeline. The current table-id must be greater than the previous table-id.	Required Instruction

Table 2.5 List of Instruction Types

The instruction set which is associated with a flow entry contains no more than one instruction of each type and the instructions of the set execute in the order specified by the Table 2.5. If a switch cannot execute the instructions it must reject the flow entry associated with these instructions.

OpenFlow Channel

The OpenFlow channel [20] is the interface that connects each switch to the controller. The controller configures and manages the switch, receives events from it, and sends packets out it, using this interface. All messages have to be formatted according to the OpenFlow protocol. This channel is often encrypted using TLS, but could run directly over TCP.

The OpenFlow protocol supports three message types - controller-to-switch, asynchronous, and symmetric. To each of these messages can be associated multiple sub-types. The first type message - “Controller-to-Switch” are initiated by the controller and is used to manage and/or inspect the state of the switch and may or may not require a response from it. The second kind of messages, “Asynchronous”, are sent without a controller soliciting them from the switch. Switches send asynchronous messages to controllers to report a packet arrival and/or switch state change.

The three main asynchronous message types are Packet-in, Flow-Removed and Port-status. The last message type used by OpenFlow protocol is “Symmetric messages” and they are initiated by either the switch or the controller and sent without solicitation. They include messages such as Hello, Echo, Error and Experimenter.

2.4 CONTROL PLANE

In SDN, the control plane is logically centralized and decoupled from the data plane. The SDN controller translates application requirements and controls the SDN data paths, while communicating with SDN applications. Decisions are made with a current view of the entire network rather than within the limited visibility of each network hop, as routers do today. The SDN controller is essentially a network operating system that constructs and presents a logical map of the network to services or applications that are implemented on top of it.

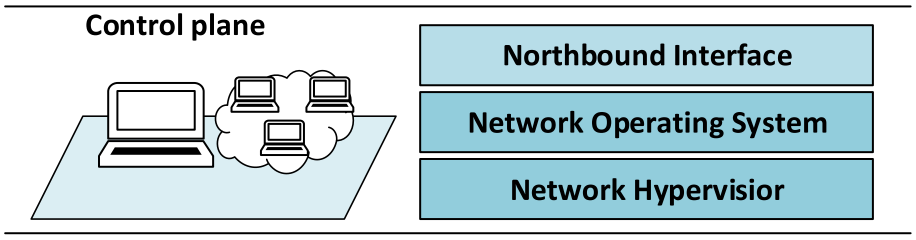


Figure 2.6 SDN Control Layer

Control plane generally includes Network Hypervisors Layer and Network Operating System (NOS), which can be seen on Figure 2.6.

Network Hypervisors Layer

Virtualization layer helps in the development and operation of SDN Slice on the top of shared network infrastructure.

In the past decade one of the mainstream became the virtualization of computing platforms. Based on recent report the number of virtual servers has already exceeded the number of physical. However to provide the complete virtualization, the network should provide similar properties to the computing layer. Long standing virtualization primitives such as Virtual Local Area Networks (VLANs), NAT and Multiprotocol Label Switching (MPLS) are based on a box-by-box basic configuration so there is no chance to configure the network in a global manner. But there is a hope that with SDN and the availability of new tunnelling techniques the situation will be change. Solutions like FlowVisor[21], FlowN[22], AutoSlice[23] have been recently proposed, evaluated, and deployed in real scenarios for on-demand provisioning of virtual networks.

FlowVisor is one of the earliest technologies to virtualize an SDN, which enables network virtualization by dividing a physical network into multiple logical networks. FlowVisor ensures that each controller touches only the switches and resources assigned to it. It also partitions bandwidth and flow table resources on each switch assigns those partitions to individual controllers. Each slice receives a minimum data rate, and each guest controller gets its own virtual flow table in the switches. Five slicing dimensions are considered in FlowVisor: bandwidth, topology, traffic, device CPU, and forwarding tables. FlowVisor is a transparent proxy that intercepts OpenFlow messages between switches and controllers and enables multiple controllers to operate the same physical infrastructure. Other OpenFlow controllers then operate their own individual network slices through the FlowVisor proxy. This arrangement allows multiple OpenFlow controllers to run virtual networks on the same physical infrastructure.

OpenVitreX [24] operates like a proxy between the Network Operating System and the forwarding devices likewise OpenVisor. By exposing OpenFlow networks, OpenVirteX allows tenants to use their own network OS (NOS) to control the network resources corresponding to their virtual network. OpenVitreX creates multiple virtual software defined networks out of one. Unlike FlowVisor, which simply slices the entire flow space amongst the tenants, OpenVirteX provides each tenant with a fully virtualized network featuring a tenant-specified topology and a full header space.

AutoSlice is another proposal of SDN-based virtualization. It enables substrate providers to resell their SDN to multiple tenants while minimizing operator intervention. AutoSlice aims also scalability aspects of network hypervisor by optimizing resource utilization and by mitigating the flow-table limitations through a precise monitoring of the flow traffic statistics. Similarly to AutoSlice, AutoVFlow also enables multi domain network virtualization. Moreover, allows a flow space virtualization to be implemented without the need of a third party in an autonomous way by exchanging information among the different domains.

FlowN is based on a different principle performs lightweight container-based virtualization, instead of running a separate controller per tenant like in FlowVisor. The FlowN architecture provides each tenant with the illusion of its own address space, topology, and controller. The FlowN controller platform leverages database technology to efficiently store and manipulate mappings between the virtual networks and the physical switches.

The basic target of SDN hypervisor is to allow sequential or parallel implementation of application developed with different programming languages or conceived for diverse control platforms. It thus offers interoperability and portability in addition to the typical functions of network hypervisors.

None of the above mention virtualization approaches are address to all challenges of multitenant data centres. VMware has proposed a network virtualization platform (NVP) that provides the necessary abstractions to allow the creation of independent virtual networks for large-scale multitenant environments. NPV allows the creation of virtual networks over the same physical network each with independent service model, topologies and addressing architecture. Using NPV tenants do not need to know anything about the underlying network. A cluster of SDN controllers are used by the platform to manipulate the forwarding tables of the Open vSwitches in the host's hypervisor. The packet is tunnelled after the decisions are made over the physical network witch sees only IP packets to the receiving host hypervisor.

IBM has recently offer SDN VE which architecture is built on open standards OpenDaylight. This allows a complete implementation framework for network virtualization. SDN VE use a host-based overlay approach like NVP achieving advanced network abstraction that enables application-level network services in large-scale multitenant environments.
In addition, there are already a few network hypervisor suggestions coping with the advance of SDN, but there is still several problems to be addressed. Among others, these include the improvement of virtual-to-physical mapping techniques, the definition of the level of detail that should be exposed at the logical level, and the support for nested virtualization.

Network Operating System/Controllers

NOS or a controller is a crucial element in an SDN architecture that manages the flow control to enable intelligent networking. It is the strategic control point in the SDN network, relaying information to the switches/routers 'below' (via southbound APIs) and the applications and business logic 'above' (via northbound APIs). The controller acts as an arbiter, abstracting the underlying physical network topology from the applications that wish to program them. SDN controllers are based on protocols, such as OpenFlow, that allows several servers to tell the switches where to send the packets.

Existing SDN controllers can be categorized based on many aspects. Looking from the architectural point of view, one of the most corresponding is if they are classified by centralized or distributed controllers.

Centralized controllers represent a single entity that manages all forwarding devices of the network. It is sees and knows everything about the network, including where the hosts connect to the network and what the network topology connecting all of the hosts together look like. But a large number of data plane elements it may could not be manage by a single controller. For example, centralized controllers such as Maestro, Beacon, NOX-MT and Floodlight have been designed to achieve the throughput required by enterprise class network centres. Others such as Ruy NOS, Meridian and ProgrammableFlow aim particular environments like cloud infrastructure, data centres. A controllers such as Rosemary distinguishes itself by its blend of process containment, resource utilization monitoring, and an application permission structure, all designed to prevent common failures of network applications from halting operation of the SDN Stack by using a container-based architecture called micro-NOS.

Turning to the ***distributed controllers*** they can meet the requirements of potentially any environment, from small to large scale networks. They can be a centralized cluster, which propose high throughput for very dense data centres or a physically distributed set of elements more resilient to different kinds of logical and physical failures. Examples of distributed controllers are Onix, HyperFlow, DISCO, SMaRT-Light and etc. One key limitation of the distributed controllers is that the mapping between a switch and a controller is statically configured, which may result in uneven load distribution among the controllers. A distributed controller can improve the control plane resilience and scalability by reducing the impact of problems caused by network partition, for example.

Eastbound and Westbound APIs are a special case of interfaces which are needed by distributed controllers. Each controller have its own east/westbound API, the functions of these interfaces involves exchanging data between controllers, algorithms for data consistency models, and monitoring /notification capabilities. East/Westbound APIs in multi domain setup may will need more specific communication protocols between SDN domain controllers. Some of the main functions of such protocols are to exchange reachability information to facilitate inter-SDN routing, coordinate flow setup occurred by the applications, reachability update to keep the network state consistent, among others. Heterogeneity is another important problem when it comes to east/westbound interfaces. For example, controllers may need to communicate with subordinate controllers or non- SDN controllers, besides the communication with peer SDN controllers. “SDN compass” methodology offers more delicate distinction between the both horizontal interfaces, referring to westbound interfaces as SDN-to-SDN protocols and controller APIs while eastbound interfaces would be used to refer to standard protocols used to communicate with legacy network control planes.

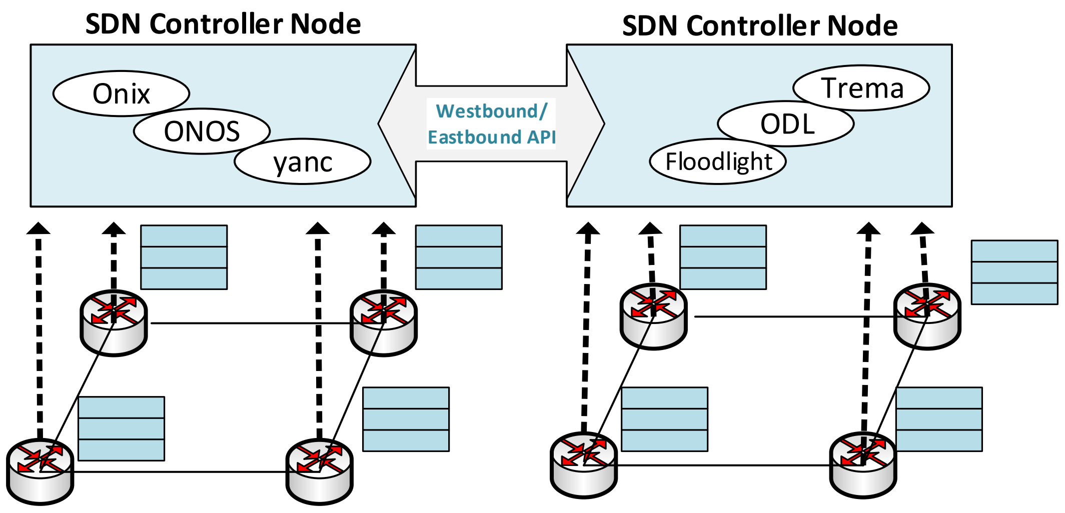


Figure 2.7 Eastbound and Westbound APIs in SDN

2.5 APPLICATION PLANE

The application layer is placed above the control layer. SDN applications can access a global network view with instantaneous status via a northbound interface of controllers [3]. For instance, the Application Layer Traffic Optimization (ALTO) protocol [25], [26] and the eXtensible Session Protocol (XSP) [27]. Using the obtained information, the applications can programmatically implement methods to manipulate the underlying physical networks. That can be performed using a high level language which is provided by the control layer. From that perspective, SDN brings "Platform as a Service" model for networking [28]. In the following subsection, different SDN applications built on this platform are described.

2.5.1 ADAPTIVE ROUTING

Packet switching and routing can be considered as the main functions of current networks. Usually, their design is based on distributed approaches for robustness. Unfortunately, such distributed design has many drawbacks, such as complex implementation, slow convergence [29], and limited ability to achieve adaptive control [30]. In contrast, SDN networking brings a closed loop control. The applications now have access to instantaneous global network status information and that allows them to adaptively control a network. Next, a description of two well-known SDN applications that utilize the SDN platform for better routing designs has been done.

Load Balancing

This is extensively used approach to achieve better usage of the resources. A wide used practice to increase throughput, decrease response time, and avoid overloading of network is the use of front-end load balancers in data centres. Unfortunately, these load balancers are very expensive. SDN can provide another alternative approach to solve that. Examination of methods to balance load using packet and uses cases in several scenarios are presented in the next part of this work.

In [31] an initial efforts to build a model using packet forwarding rules of SDN and develop approaches for load balancing has been made. In this paper the authors assume homogenous traffic from all clients with different IP addresses and suggest a usage of a binary tree to arrange IP prefixes. Using wild card rules, the traffic is divided. A server replica will handle a traffic whose size is in proportion to its processing ability. In most of the cases the assumption they made could not be true. We can consider this work as a basis for future work on load balancing that can take advantage of packet forwarding rules of SDN. Another research [32] proposed different approach. The authors migrate the traffic from heavy loaded switching devices to lightly loaded ones reactively. While algorithm development is crucial, efforts in deploying load balancing with SDN in various scenarios has been made. Several services may require their own specialized load balancing algorithm implementations and do not want to affect others, even if some load balancing implementations break down. In consideration of that, in [33] has been introduced a load balancing method for different kinds of traffic (for instance, web traffic and email traffic). Their aim is to provide dedicated and specialized balancing algorithm based on the requirements of services and workloads. Another case has to be mentioned, when a front-end load balancer has to direct every request and may become the bottleneck of a data centre. In [34] the authors tried to solve this issue. They presented Aster∗x, which balances load over an arbitrary unstructured network. To decrease the response time of web services a direct control of the paths taken by new HTTP requests has been used.

Cross-Layer Design

This approach is a widely used method for enhancing integration of entities at different layers in a layered topology (for instance, OSI model). Cross-Layer approach allows entities at different layers to transmit information among each other. These methods can be easily implemented on SDN platform, because it provides to application easy and efficient access to network status data. In this subsection, use cases to provide QoS and enhancing application performance using the cross-layer design method are presented. A lot of network applications need a certain level of QoS support. An effective cross-layer method to achieve guaranteed QoS is based on QoS information for appropriate network resource reservation. In [35] the authors provide enhanced QoS for video conferencing. They retrieve schedule of available bandwidth from an SDN controller and the earliest time when a video or an audio call can be made is calculated. Another work [36] presents QoS-aware Network Operating System (QNOX) to provide QoS guaranteed services. Some of them are QoS-aware virtual network embedding and end-to-end network QoS assessment. QNOX is making QoSaware mapping for the virtual network on the substrate network for a request and also monitors end-to-end QoS providing calculated results to enhance making of operational changes. Furthermore, the adaptive routing may improve the performance of the applications. Wang et al. [37] suggest a cross-layer method to configure underlying network at runtime depending on big data application dynamics. They has been took advantages of high reconfigurability of SDN switches and also the high speed and reconfigurability of optical ones. They use Hadoop job scheduling approaches to accommodate dynamic configuration of the network in a hybrid network with Ethernet and optical switches. The results of their analysis are improved performance of the application and network utilization with approximately low overhead of the configuration.

2.5.2 BOUNDLESS ROAMING

Mobile devices, such as smartphones and tables, have wireless access to the Internet and they are the most widely used devices in this environment. They required a continuous connectivity because they are not static, and for that reason these connections have to be carried over from one base station to another (sometimes from one wireless network to another). It is obvious that the handover has to provide uninterrupted services. Nowadays, the handover is limited to networks of a single carrier with the same technology, but SDN can provide a common control plane for networks of different carriers with are using different technologies. In that perspective, some SDN handover schemes have been proposed in the literature. In [38]-[40], Yap et al. suggest handover approaches between Wi-Fi and WiMAX. They use multiple interfaces on a device and n-casting, which duplicates traffic across n distinct paths. In order to decrease packet loss and improve TCP throughput during handover this methods could be implemented with SDN without much effort. In [41] the authors proposed a prototype SDN framework for enterprise WLANs. It is called Odin and it relies on allocation of a unique Basic Service Set Identification (BSSID) for each connected client. The main idea is to performing handover based on removing of the BSSID from one physical wireless Access Point (AP) and spawning it to another AP. The authors achieved low delay in re-association, no throughput degradation and minimum impact on HTTP download.

2.5.3 NETWORK MAINTENANCE

Configuration errors can be seen as common and severe cause of network failures. According to [42] it can be said that more than 60% of network downtime is a result of human configuration errors. Furthermore, existing network tools that handle with individual diagnosis, fail to provide an automated and comprehensive network maintenance approaches. SDN can be bring solution and solve the problem with configuration errors, because of its centralized and automated management and coherent policy enforcement. In addition, a new design for comprehensive network diagnosis and prognosis mechanisms for automated network maintenance can be achieved due to the global view and central control of configuration in SDN. To detect the causes of network failure a network diagnosis tool is required. Examples of these tools are ndb [43] and OFRewind [44]. The main idea of the first approach is to provide back-trace of network events to the controller, when a packet visits a switch. The back-trace is built by the controller to debug the network. Similarly, the second tool constantly records network events in the network, but in addition it replays these events later to debug the network. The SDN central control may directly resolve network failures and the routing convergence time is shorter [45]. In [46] a fast restoration method for SDN networks is has been presented. It relies on calculation of new forwarding paths for affected ones and immediately updates packet forwarding rules.

2.5.4 NETWORK SECURITY

Nowadays, network security is gaining attention as a part of the cyber security. Classic network security approaches uses firewalls and proxy servers to protect the physical network. Because of the heterogeneity in network applications, providing of exclusive accesses by legitimate network applications requires deployment of a network-wide policy and complicated configuration of firewalls, proxy servers, and/or other devices. SDN brings a suitable platform to centralize, merge and check policies and configurations, in order to ensure good implementation of protection and to prevent security problems proactively. Furthermore, SDN uses better approaches to detect and even defend attacks reactively. The collecting of network status can be used again to perform analysis of traffic patterns for potential security problems. For example, attacks like low-rate burst attacks and Distributed Denial-of-Service (DDoS) attacks, can be noted in the network only by analysing traffic pattern [47], [48]. Moreover, detected traffic can be directed to Intrusion Prevention Systems (IPSs) for Deep Packet Inspection (DPI), using the function of SDN networks that allows a programmatic control over the flows [49], [50]. In case of detection of malicious traffic, SDN may install packet forwarding rules to its switching devices in order to block the attack [51]. A lot of research shows that SDN, due to its centralized control, is able to provide dynamically quarantine of compromised hosts and authentication of legitimate hosts, requesting a Remote Authentication Dial In User Service (RADIUS) server for users' authentication information, tainting traffic or system scanning during registration. Moreover, SDN is also able to provide direct and fine-grained control over networks, and brings the opportunity to implement innovative security protection methods. For instance, in [52] the authors have been made Moving Target Defence (MTD). The main idea is to associate a virtual IP with each host for data transmission. It is modified unpredictability while a real IP address of the user is static. Controllers are defining the translation between the virtual IP and real IP and maintaining the integrity of configuration. In [53] is presented AnonyFlow service designed to provide privacy to users. This approach makes translation between AnonID, Network IP and Machine IP based on SDN.

2.5.5 NETWORK VIRTUALIZATION

Network virtualization is a well-known method to allow multiple heterogeneous network architectures to work together on a shared infrastructure [56]. A popular practice of network virtualization is to divide the physical network into multiple virtual entities and after that to assign them to different users, controllers, or SDN applications. This can be seen on Figure 2.8.

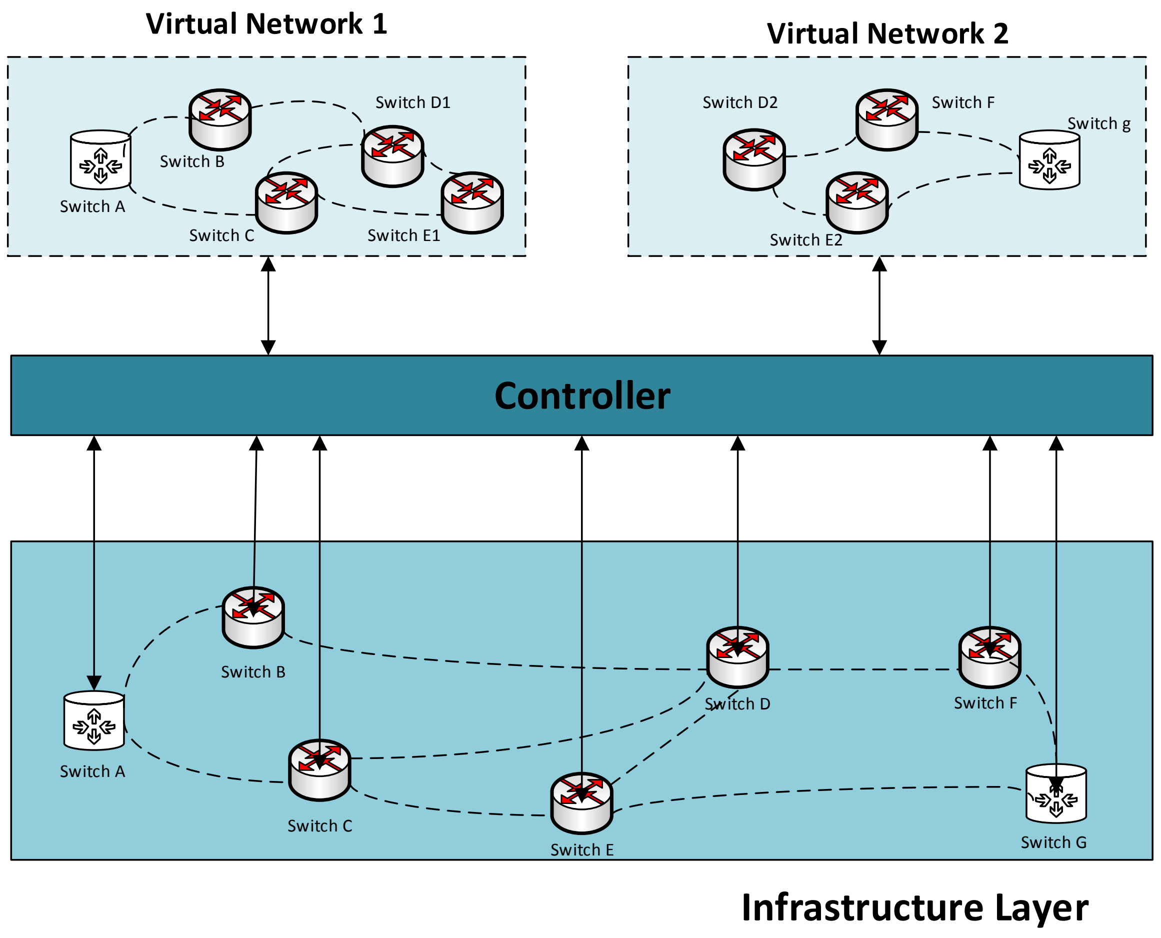


Figure 2.8 Network virtualization example

Standard virtualization approaches that use tunnels and VLAN or MPLS tags need difficult configurations on all network devices that are participle in it. In contrast, SDN provides a platform that allows configuration of all switching devices in a network from a single controller. Example of that virtualization method is libNetVirt [57]. Using this platform, several strategies can be deployed at the application layer to automate configuration for slicing the network resources. FlowVisor [21] is another example of how these resources can be divided, including bandwidth, topology, flow space, switches CPU, forwarding table space, and control channel. FlowVisor is acting as a transparent proxy to filter control messages and it is placed between the guest controllers and the switching devices. FlowVisor can be used to create virtual networks from a physical network for research experimentations and also to share a physical network with several users with good isolation. In [58] another method for network virtualization by bringing isolation at a language level is presented. It is based on taking as an input a collection of slice definitions and their associated applications. After that for each switch is generated a list of packet forwarding rules to create a convenient virtual network.

2.5.6 GREEN NETWORKING

Nowadays, the idea of designing and development of "Green" networks takes a significant attention. It brings economic and environmental benefits. Several methods have been already proposed in [54] to achieve green networking, such as energy-aware data link adaptation, energy-aware traffic proxying, energy-aware infrastructure and energy-aware application. SDN switch does not propose directly energy reduction, but SDN could offer huge promises in minimization of network-wide energy consumption. In [55] the authors proof that with demonstration of energy-aware data link adaptation with SDN network. They have been proposed an approach to determine minimum data links and switching devices for a data centre network. This method relies on traffic loads and dynamically power down redundant links and switching devices for energy efficient operations.

2.5.7 SDN FOR CLOUD COMPUTING

Cloud computing is relatively new way of doing computing and business and it relies on computing and storage resources on demand. In addition it charges on usage with server and network virtualization. SDN comes with a lot of ways to expand the service provisioning model of Infrastructure as a Service (IaaS) and includes a huge set of additional network services for more flexible and efficient cloud computing [59]. Data centre networks for cloud computing have a few main needs. According to [60] we can mention some of them, namely scalability for large scale deployment, location independence for dynamic resource provision, QoS differentiation for several tenants, and network visibility and fine-grained control. SDN architecture can absolutely satisfy the requirements of the data centre networks for cloud computing. Next in this subsection discussion about virtual switching and virtual machine (VM) migration has been done. They are two of the main issues in cloud computing.

Virtual switching

Virtual switching relies on a software program that allows one virtual machine to communicate with another, but unfortunately, the well-known hypervisors that provide virtual switching does not meet the requirements of visibility and control. To meet these needs Open vSwitch comes with virtual edge switching for VMs with visibility and control [61]. It uses the idea of SDN. These special types of switches have also the same functions as a conventional SDN switching devices and report network status to and receive packet forwarding rule, but they provide limited storage and processing resources compared with physical switches. In [62] a virtual Cloud Rule Information Base (vCRIB) is proposed and it can solve these resource issues. It automatically finds the optimal rule placement among physical and virtual switches. At the same time it minimize traffic overhead and can quickly adapt to cloud dynamics (for instance, traffic changes and VM migrations).

VM migration

According to [63] it can be said that VM migration is widely used in data centres for statistical multiplexing or dynamic communication pattern changing. The aim is achievement of higher bandwidth for tightly coupled hosts. While IP addresses have to be preserved across broadcast domains, the standard VM migration is usually limited to a single broadcast domain and the Address Resolution Protocol (ARP) messages cannot go beyond this domain. There is some research in the direction of Mobile IP and Locator/Identifier Separation Protocol (LISP) and based on that some SDN solutions has been proposed. For example, [64] proposes OpenDaylight SDN controller with LISP-based mapping service. In [32] the authors claim that SDN can support VM connectivity during intra and inter data centre VM migration using proper packet forwarding rules in switches to direct traffic to the new VM location. In [65] is proposed CrossRoads, it is an example of inter data centres VM migration. CrossRoads, using SDN implementation, provides migration of virtual machines across data centres. It expands the concept of location independence. CrossRoads is based on pseudo addresses proposed in cloud networking to work with controllers governing their own data centre network. After the inter data centre migration, ARP messages are used to recognise an IP address outside a subnet. VM connections can be supported during inter data centre migration by sending packets with previously recognised IP address to their destination in an external data centre.

3 VEHICULAR AD HOC NETWORKS VANETS

3.1 VANET DEFINITION

VANETs (Vehicular Ad Hoc Networks) is also known as Vehicle-to-vehicle (V2V). This technology is based on of MANETs (Mobile Ad Hoc Networks), where the node is no more mobile device but instead a vehicular. Examining of MANET is crucial to understand V2V, since MANET is the basis of VANET.

3.1.1 MOBILE AD HOC NETWORKS (MANETS)

With the wide spread of mobile devices such as mobile phones, laptops, personal digital assistants (PDA) and etc., combined with the significant increase in the wireless sector in the last decade, the way the information is processed undergoes considerable change [66]. Consumers carry mobile equipment that provide network services and run applications, from which the most desired one by the users is data services. Based on infrastructure are most of the connections between mobile devices. For instance, mobile phone communicate with each other via cell phone towers, two or more portable computers are connected using a wireless access point. To establish a mobile device communication infrastructure is potentially costly. Sometimes consumers could meet instances where the infrastructure demand for particular communication is simply not reachable. Furthermore devices such as PDAs and laptops have only short range wireless ability. This causes the evolution of various ways of mobile communication where the nodes of the mobile devices interacts with each other over wireless without any infrastructure support, forming a mobile ad hoc network.

Irrespectively from centralized administration or any infrastructure nodes in MANET works in a peer to peer mode. Intermediate nodes send messages to destination node over multiple hosts so it can communicates with nodes which are beyond the range. The nodes and the data generated by them are independent in this network environment. Network topology can change frequently and unexpectedly therefore that the nodes in MANET are mobile. Nodes in the wireless diapason quickly find each other and determine connection with each other, because MANETs do not rely on any form of control or central administration. This helps to support the ad hoc network even in cases where nodes continue moving in an out of each other wireless range.

Both legitimate and illegitimate users can access the shared wireless medium. In compression to the fixed line networks possibilities of spoofing, denial of service attacks, eavesdropping are more prevalent [67]. The applications including MANETs range from those that include small number of nodes to those comprising tens of thousands of nodes.

Table 3.1 depict some of the current and future radio technologies for MANET applications [68].

Technology	Bit Rate (Mbps)	Frequency	Range
802.11b	Upto 11	2.4 GHz	25-100 m (indoor)
802.11g	Upto 11	2.4 GHz	25-50 m (indoor)
802.11a	Upto 11	5 GHz	10-40 m (indoor)
802.15.1	1	2.4 GHz	10 m (up to 100 m)
802.15.3	110-480	3-10 GHz	~10 m
HiperLAN2	Upto 11	5 GHz	30-150 m
IrDA	Upto 11	Infrared (850nm)	~10 m (line of sight)
HomeRF	1-10	2.4 GHz	~50
802.16	32-134	10-66 GHz	2-5 km
802.16 a/e	15-75	<6 GHz, <11GHz	7-10 km, 2-5 km

Table 3.1 Wireless technologies in MANET

3.1.2 *VEHICLE-TO-VEHICLE COMMUNICATIONS /VEHICULAR AD HOC NETWORKS (VANETS)*

According to most of the sources V2V communication defines as a type of network in which road side units (RSU) and vehicles are nodes which communicate and provide information with each other, like traffic information and safety warnings. As a joint approach vehicular communication systems can be more efficient in preventing traffic congestions and accidents than if each vehicle tries to individually solve these problems.

Typically, vehicular networks have to types of nodes: vehicles and roadside stations. They are devices using dedicated short-range communications (DSRC). DSRC works in 5.9 GHz band with bandwidth of 75 MHz and range of approximately 1000 m, it is an Ad Hoc communication, which means there is no wires needed and each node which is connected can move freely. Routers are used to called RSU, since they operates like a router between the vehicles on the road and interacted to other devices network. OBU (on board unit) are placed in each vehicle, and they via

DSRC radios connects the RSU with vehicle. Public communication obtains higher priority compared to private data communication nevertheless that the network should support both. Vehicular communications is commonly evolved as a part of intelligent transportation systems (ITS). ITS represents advanced applications which are intended to ensure modernistic services linked to different modes of transport traffic management and allow various users to be informed more properly and make more coordinated and safer the use of transport networks.

IEEE 802.11 is the standard on which the communication technology is based, aka as Wireless LAN. A frequency spectrum in the 5.9-GHz range has been distributed on a harmonized basis in Europe in line with similar allocations in the U.S. (although the systems are not yet compatible).

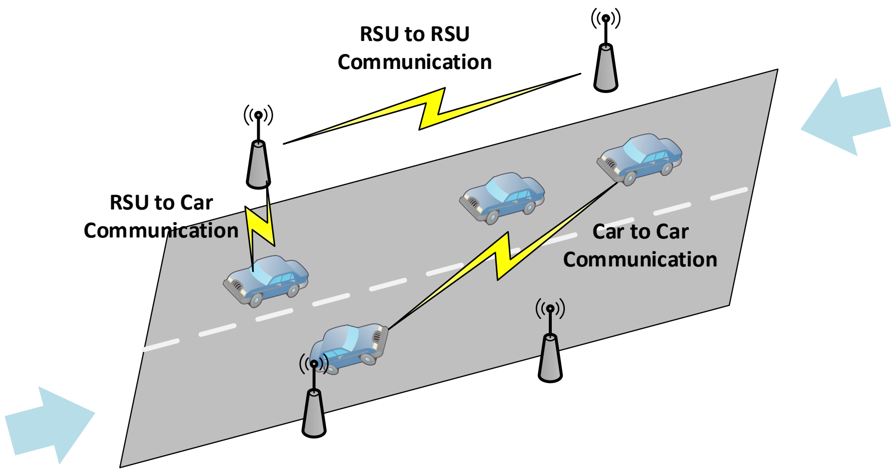


Figure 3.1 VANET Structure

3.2 IEEE 802.11 STANDARD

The IEEE 802.11p Wireless Access in Vehicular Environments (WAVE) standardization process derived from the distribution of the DSRC spectrum band in the United States and the effort to determine the technology for usage in the DSRC band.

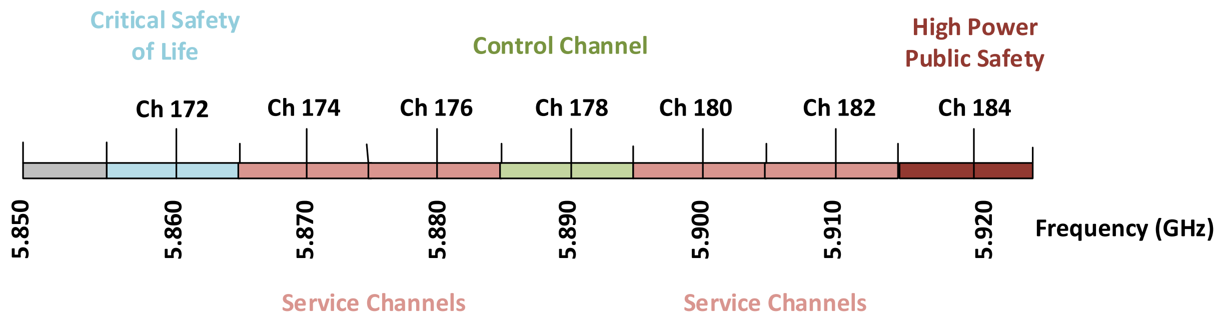


Figure 3.2 DSRC spectrum band and channels in the U.S.

The DSRC spectrum is structured into seven 10 MHz wide channels as it shown in Figure 3.2. Control channel (CCH) is the channel 178, which is limited only to safety communications [69]. For special purposes are reserved the two channels at the ends of the spectrum band.

IEEE 802.11p standard or WAVE is specially developed to adjustment the VANETs requirements and support ITS. It has been designed to addresses the needs of connecting wireless devices in a rapidly changing networking environment and in cases when in shorter amount of time the data transactions should be completed.

The IEEE 802.11p standard is intended to:

- Describe the functions and services demanded by WAVE-conformant stations to work in a rapidly changing environment and exchange messages without having to join a Basic Service Set (BSS), as in the traditional IEEE 802.11 use case.

-Determine the WAVE technique for signalization and interface functions that are commanded by the IEEE 802.11 MAC.

Figure 3.3 illustrates DSRC standards and communication stack.

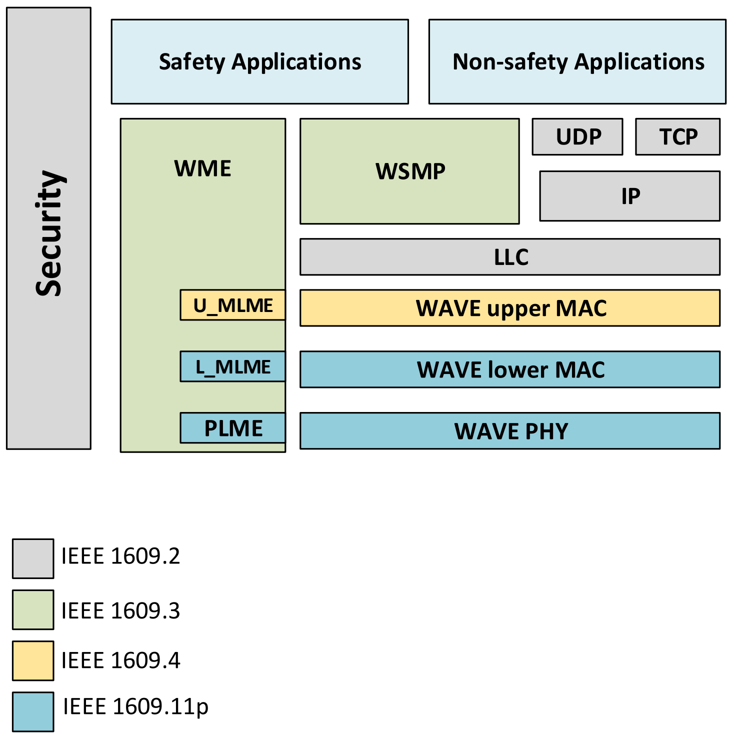


Figure 3.3 DSRC standards and communication stack

IEEE 1609.x and IEEE 802.11p together formed the WAVE architecture as shown in Figure 3.4.

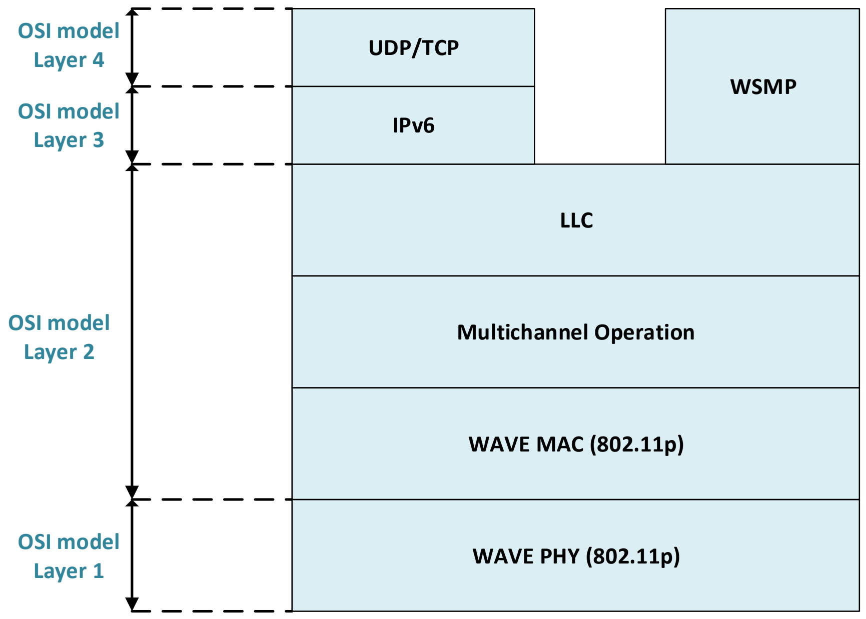


Figure 3.4 WAVE Communication Protocol Stack

3.2.1 *MAC AMENDMENT DETAILS*

In general case, the IEEE 802.11 MAC aims to organize a set of radios in order to determine and maintain a cooperating group. Within the group radios can freely communicate between themselves but all the transmissions from outside are filtered out. Example for such a group is a BSS a lot of protocol mechanisms projected to provide robust and secure communications within it. The main goal of the IEEE 802.11p variation at the MAC level is to allow efficient communication group setup without much of the overload, which in most cases is needed in the present IEEE 802.11 MAC. The aim is to simplify the operations in BSS in an actually ad hoc manner for vehicular usage.

An Infrastructural Basic Service Set is a group of IEEE 802.11 stations anchored by an AP and configured to communicate over air connection among each other. In most cases it is just referred to as a BSS. The BSS mechanism command the entrance to an AP’s services and resources, and also permits radio to filter out the transmissions from other unconnected radios which are close. The first step that radio do is to listens for beacons from an AP and after that connects the BSS through different steps, such as association and authentication.

As depicted in Figure 3.5, the IEEE 802.11 standard additional permit administrators to join logically group of one or more interrelated BSSs into one ESS (Extended Service Set) by the use of DS (Distribution Service). An ESS arises as a single BSS to the LLC (Logical Link Control) layer at any station correlated to one of those BSSs.

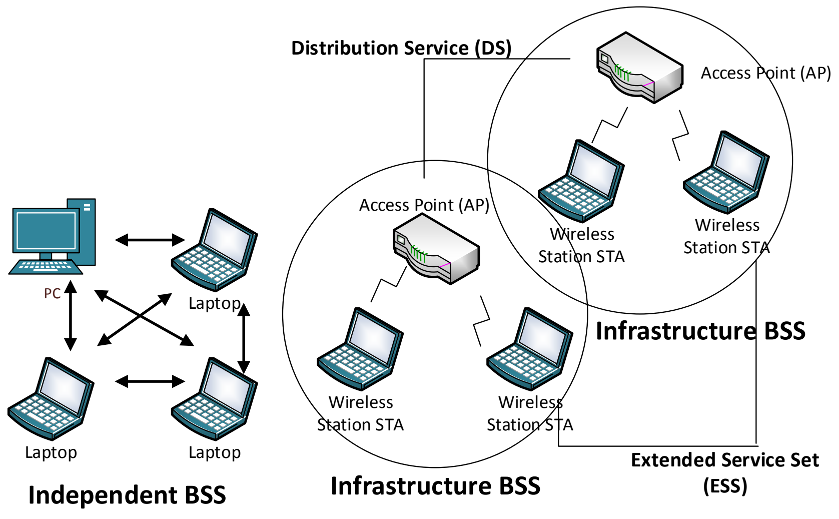


Figure 3.5 Independent and extended service set concepts

Due to the Service Set Identification (SSID) BSS is familiar to the users. This corresponds to the names of WiFi hotspots that users can look on and connect to at public locations or home. In the range of 0 and 32 Bytes is the information field of SSID. It should be mention that SSID is differ from BSSID, which stands for Basic Service Set Identification. BSSID is the name of a BSS familiar to the radios at the MAC level and exactly like a MAC address it has a 48-bit long field. Each BSS necessarily need to have a unique BSSID shared by all the members. With help of the MAC address of the AP this could be easily guaranteed.

A locally administered IEEE MAC address is used for an IBSS. This is composed through the use of a random number with the individual/ group bit set to 0 and the universal/local bit set to 1.

BSSID filtering is crucial mechanism for restricting all incoming frames to only those radios which are part of the same BSS, at the MAC level.

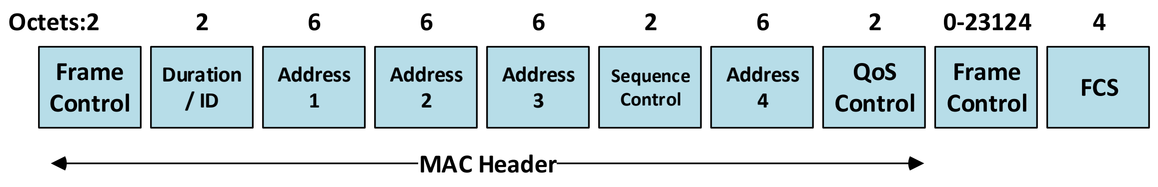


Figure 3.6 IEEE 802.11 data frame format

Each IEEE 802.11 data frame contains up to 4 address fields, as it depict in Figure 3.6. Address field are used to bring source address (SA), destination address (DA), address (TA), transmitting STA, receiving STA address (RA) and BSSID. As shown in Table 2.2 the use of four address field in the frame control field varies depending to the "To DS" and "From DS" bits.

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	RA = DA	TA = SA	BSSID	N/A
0	1	RA = DA	TA = BSSID	SA	N/A
1	0	RA = BSSID	TA = SA	DA	N/A
1	1	RA	TA	DA	SA

Table 3.2 IEEE 802. 11 data frame address field contents

On receipt a frame from the PHY, the MAC level of a station, uses the contents of the Address 1 field to execute address matching for receiving decisions. In case Address 1 field includes a group address, the BSSID is compared to secure that the broadcast or multicast comes from a station in the same BSS. Wildcard BSSID is a special case of the BSSID, and it is made of all “1s”. The use of wildcard is restrict from the present IEEE 802.11 standard.

IEEE 802.11 MAC operations are still not added in IEEE 802.11p due to the fact that they are too time-consuming, as described above. Since the cases of safety vehicular communications require immediate data exchange capabilities and cannot afford scanning channels for the beacon of a BSS and then performing multiple handshakes to determine the communications.

WAVE MODE

The term “WAVE mode” is a key change entered by the IEEE 802.11p WAVE. When a station is in WAVE mode, it may permit to receive and transmits data frames by the use of wildcard BSSID value without the necessity of any kind a priori or belonging to a BSS. Put in another way, two vehicles as long as they work in the same channel using the wildcard BSSID, they can immediately communicate with each other without any additional overload.

WAVE BSS

The overload of traditional BSS setup may be it is going to be too expensive even for non-safety services and communications. For instance, a vehicle approaching a road side station that offers, say digital map download service, can hardly afford the many second needed normally in a conventional WiFi connection setup since the total time the vehicle will spend in this range is very short. A new BSS type WBSS (WAVE BSS) was presented by the WAVE standard. A station represents a WBSS by first transmitting upon require beacon. The require beacon is used from the WAVE station, it uses the well- known beacon frame and should be not repeated periodical, to advertise a WAVE BSS. Such an advertisement is created and consumed by upper layer mechanisms above the IEEE 802.11. It includes all the needed information for receiver stations to known the services offered in the WBSS and decide whether to join, and also the information needed to configure itself into a member of the WBSS.

Put in other way, a station by only receiving a WAVE advertisement can decide to join and complete the joining process of a WBSS with no further interactions. By eliminating all authentication and association processes this approach offers extremely low overload for WBSS setup. It necessitates further mechanisms at upper layers to manage the WBSS group usage as well as providing security. The mention mechanisms, however, are out of the range of the IEEE 802.11.

Expanding wildcard BSSID usage

Pointing at safety as a major factor of WAVE, even for a station that yet belong to a WBSS the use of wildcard BSSID is also supported. In order to achieve all neighbouring station in case of safety problems a station in WBSS is still in WAVE mode and can still transmit frames with the wildcard BSSID. Likewise, a station which is already in a WBSS and accordingly configured its BSSID filter, by the use of wildcard BSSID can still receive frames from others outside the WBSS. The possibility of send and receive data frames with wildcard BSSID can bring more additional benefits including safety communications. It is also capable to support signalization in present upper layer protocols in this ad hoc environment.

Distribution Service

In WAVE devices DS can be still found. Over the air, this means that it is possible for data frames to be transmitted with “To DS” and “From DS” bits determine to 1. Nevertheless, the capability for a radio in a WAVE BSS to send and receive data frames with the wildcard BSSID brings difficulties. In general case a radio will be limited to send a data frame by the use of wildcard BSSID only if the “To DS” and “From DS” bits are determine to 0. In order to reach a DS access radios which communicates within the context of a WAVE BSS need to send data frames using a known BSSID.

3.2.2 PHY AMENDMENT DETAILS

The interface between the MAC layer and the media, which gives the permission for receiving and sending frames is represented the physical layer (PHY). It is responsible for hardware specification, bit conversion, data formatting and signal coding. The IEEE 802.11p PHY is very similar to that of IEEE 802.11a. As it shown in Figure 3.7 it composed of two sub layers.

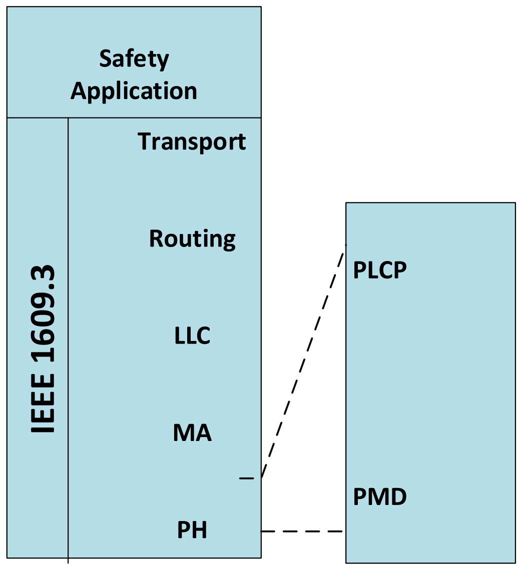


Figure 3.7 WAVE Protocol Stack and the sub layers of PHY

Physical Layer Convergence Protocol (PLCP) is the first layer and it is responsible for the communication with the MAC layer. It is also a convergence process, which make changes on the Packet Data Unit (PDU) that come from the MAC layer to build an OFDM frame. Physical Medium Access (PMD) represents the second sub-layer and it is the interface to the physical transmission medium like fiber link and radio channels. As depict in Figure 3.8. Protocol Packet Data Unit (PPDU) formed of a preamble, single field and a payload component which holding the useful data.

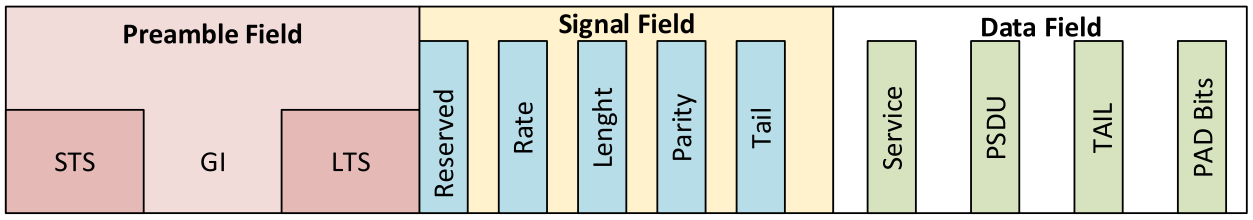


Figure 3.8 IEEE 802.11p PHY layer PPDU Frame structure

The conception of IEEE 802.11p design at PHY level is to make minimal changes to IEEE 802.11 PHY so that devices which use WAVE can communicate properly among vehicle that move fast in the roadway environment. This approach is realizable since IEEE 802.11a radios already operate at 5 GHz and it is not hard to configure the radios to operate in the 5.9 GHz band in the U.S. and similar bands internationally. PHY level changes definitely should be restricted in order to avoid the design of completely new wireless air-link technology, while the amendments at the MAC level are comparatively easy to make since they are fundamentally software updates. In the 802.11p standard important improvements that can be observed are:

10-Mhz channels

Figure 3.9 illustrates the distribution of the allocated 75MHz spectrum into 10 Mhz control and service channels. One Control Channel and four Service Channels will be support by Europe. The first channel is used for communicating Wave Short Messages and to announce WAVE services, while for application interactions the Service Channels are used. US will give support to two additional channels, High Power Public Safety channel to be used by Public Safety applications providing a better scope due to the high transmit power being used and Critical Safety of Life - to be used by SOS applications.

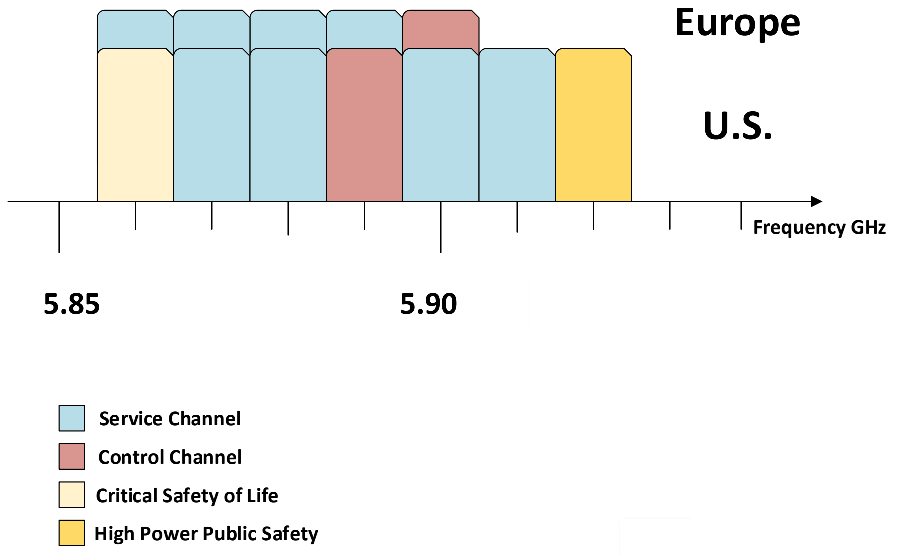


Figure 3.9 Available Channels in 802.11p

Improved receiver performance requirements

The nature of near spread vehicle on the road forms increased concern for cross channel interferences, although in the US there are a lot of available channels and (expectedly) internationally for IEEE 802.11p usage and deployment. Presented measurements in [70] show the potential for immediate neighbouring vehicles to meddle each other if they are operating in two adjacent channels. A well-known natural and physical property of wireless communications is a cross channel interference. The most proper and effective solution to such fears is by the use of channel management policies which is totally out of the scope at IEEE 802.1. However, IEEE 802.11p presents some upgraded receiver performance requirements in adjacent channel rejections. In the proposed standard two categories of requirements are presented. The first category is mandatory and generally understood to be achievable with today's chip manufacturers. The second one is optional and probably it is going to be more expensive to perform in the next few years.

Improved transmission mask

In IEEE 802.11p defines four spectrum masks (for class A to D operations) for the specific needs of the radios in the ITS band used in U.S. (i.e., the 5.9 GHz DSRC spectrum).

3.3 APPLICATIONS OF VANETS

Application for vehicular network demand wireless communication network there are many available ways for wireless communication such as Ad hoc network cellular network, wireless LAN and info system. The network choice depends on the application demands, so we should be aware with them and their requirements.

The discovery of entirely new class of applications was imposed by VANET and their aimed to provide an improved, safer and co-operative driving experience. Here, the application can be additional divided in two types: Safety Applications which helps to increase driver's safety and Comfort Applications which take care of improving the driver's experience.

Another work consider the grouping of the applications [114]. The RSU can be viewed as an access point or router or a butter point storing and providing data when is needed. The vehicles uploaded or downloaded all data on the RSUs. Application are classified as Car to Infrastructure applications, Car to Car Traffic applications, Car to Home applications and Routing based applications.

Based on the type of communication either V2I or V2V, we are organizing the applications of VANETs into following categories: 1) Safety oriented, 2) Commercial oriented 3) Convenience oriented and 4) Productive Applications.

3.3.1 SAFETY APPLICATIONS

Applications for safety involve monitoring of the surrounding road, road surface, road curves, approaching vehicles etc. Major problems is connected with the safety time distribution of critical alerts to nearby vehicles. Applications correlated with the Road safety can be classified as:

1) Real-time traffic: At the RSU all the real time data traffic is stored and it can be accessible whenever and wherever the vehicles needed. This can play a significant role by resolving problems related to traffic jams, avoid congestions etc.

2) Co-operative Message Transfer: Slow/Stopped Vehicle will communicate by exchanging messages and co-operate in order to help other vehicle. Although reliability and latency would be a main problem, this can automate things such as potential accidents avoidances and emergency breaking.

3) Post Crash Notification: A vehicle participating in an incident will send to other vehicles warming messages about its position so they can take timely solution about their actions and as shown in Figure 3.10 to the highway patrol for toll away support.

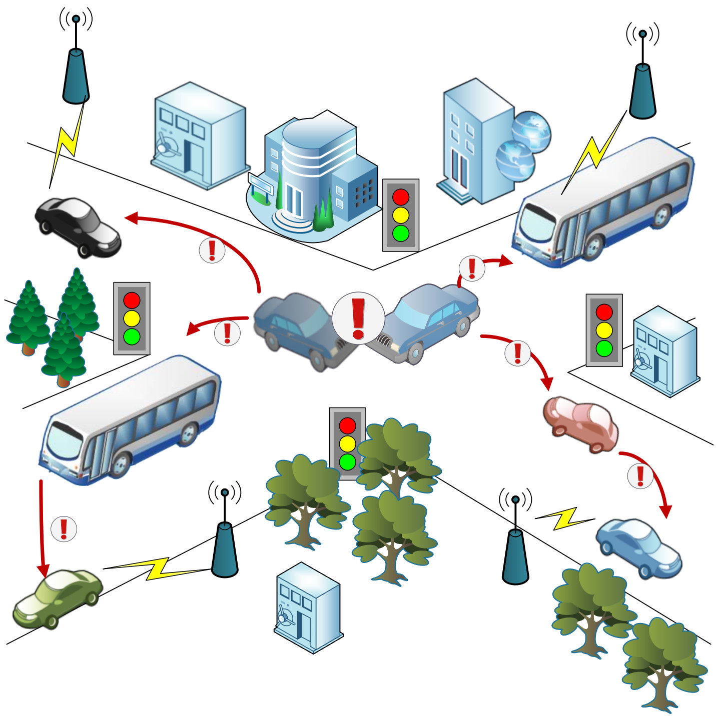


Figure 3.10 Emergency situation Notification

4) Road Hazard Control Notification: Vehicles give other vehicles information about road having landslide or road feature notification due to sudden downhill, road curve etc.

5) Cooperative Collision Warning: Notifies two drivers which have great chance to crash on the road, so they can take timely decision to change the road.

6) Traffic Vigilance: In the RSU cameras can be installed, so they can work as input and behave like the latest tool in low or zero tolerance against driving offenses.

3.3.2 COMMERCIAL APPLICATIONS

Commercial applications will give the driver with entertainment opportunities and also such as web access, streaming video and audio. They can be classified as follow:

1) Remote Vehicle Diagnostics/Personalization: This application supports the network in order to download data to personalize vehicle settings or upload of vehicle diagnostics from/to infrastructure.

2) Internet Access: If RSU is working as a router cars can easily access internet through it.

3) Digital map downloading: Drivers can downloaded the regions map. Content Map Database Download behave like a portal for getting valuable information from mobile hot spots or home stations.

4) Real Time Video Relay: Services such as on-demand movie experience will not be any more restricted only of the home, but drivers will have the opportunity to ask for real time video relay.

5) Value-added advertisement: This application is directed to the service providers that have intentions to attract customers to their stores. Messages such as highways restaurants, petrol pumps to announce their services to the drivers within communication scope. In general case this application can be reachable even without Internet connation.

3.3.3 CONVENIENCE APPLICATIONS

Convenience application mainly directed to traffic management with an aim to improve traffic efficiency by increasing the degree of driver's convenience. Convenience applications can be categorized as:

1) Route Diversions: In case of road congestions trip and route planning can be made.

2) Electronic Toll Collection: By the use of Toll Collection payment of the toll can be done electronically. A Toll collection Point must be capable to read vehicle OBU. Via GPS OBUs work and the on-board techno graph or odometer as a spare to define how far the Lorries have travelled by reference to a digital map and GSM to authorize the payment of the toll via a wireless link. This application is not only useful to the drivers but to toll operators as well.

3) Parking Availability: Messages related to the parking availability in the metropolitan cities helps to find the availability of slots in parking lots in a certain geographical region.

4) Active Prediction: It anticipates the upcoming road topography, which is supposed to optimize fuel usage by controlling the speed before starting a descent or an ascent.

3.3.4 PRODUCTIVE APPLICATIONS

They are namely productive intentionally as they come up as an addition to the above mentioned applications. They can be classified as:

1) Environmental Benefits: AERIS research program [71] is to give rise to and gain environmentally-relevant transportation data in real-time, and use these data to establish real information that help and facilitate “green” transportation choices by system transportation operators and users. With the use of a multi-modal approach, the AERIS program will work in collaboration with V2V communications research effort to better identification how vehicle data and applications that are connected may contribute to mitigate, for instance, the negative environmental impacts of surface transportation.

2) Time Utilization: If a passenger downloads his email, he can turn traffic jam into a productive task and read on-board system and read it by himself if traffic stuck. One can search in the Internet when someone is waiting in car for a friend or relative.

3) Fuel Saving: When the TOLL system application for vehicle collects toll at the toll booths without persuading vehicles to stop, the fuel around 3% is preserved, which is consumed when a vehicles as an average waits normally for 2-5 minutes.

3.4 SOFTWARE-DEFINED VANET

In section 1 of this diploma thesis, we presented SDN networking and its benefits. In order to extend that statement here we want to explore how SDN can enhance VANET networks. In [72] a comprehensive survey on how SDN can be implemented in VANET network has been made. In specific, the authors proposed an architecture operations and benefits of SDN VANET services and new functionalities to support them. According to that in this chapter we are going to describe in details how SDN can be implemented in VANET scenarios. Using the separation between the control and data plane in VANETs, network intelligence and state can be centralized logically and the infrastructure of the underlying network is abstracted from the applications. Therefore it will be possible to have highly adaptive, versatile, programmable, and scalable VANETs environments.

By using V2V communication in Ad hoc fashion vehicles can communicate between each other, and V2I communication through RSU and mobile broadband such as 4G/Long Term Evolution (LTE).

Figure 3.11 illustrates the components and communications with a typical VANET.

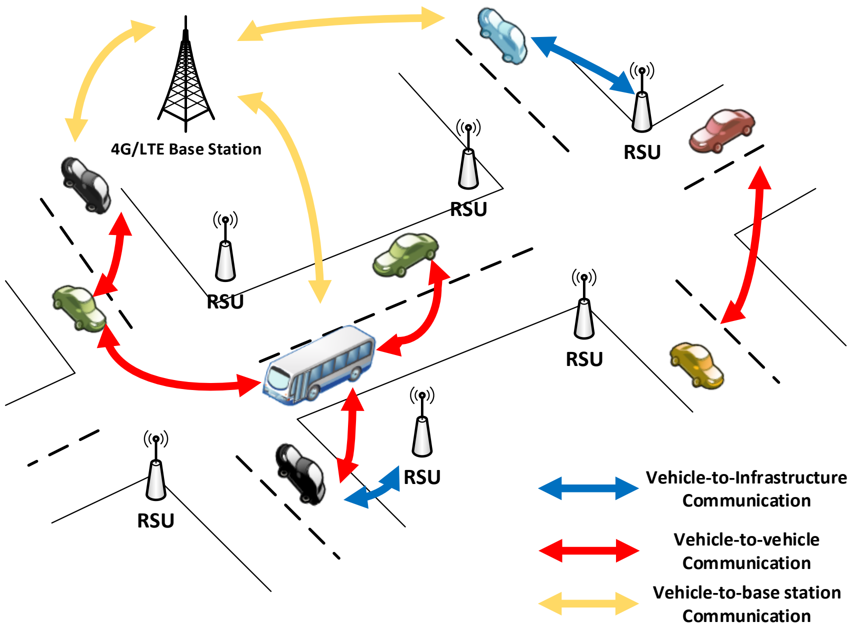

4G/LTE Base Station
RSU
RSU
RSU
RSU
RSU
RSU
RSU
Vehicle-to-Infrastructure Communication
Vehicle-to-vehicle Communication
Vehicle-to-base station Communication

Figure 3.11 VANET Component and Communications

The basis of SDN is the decoupling the control plane and the data plane. Data plane is used for data transmission, while the other is exploited for the network traffic control. The use of

SDN in environments, such as VANETs, can reduce interference: develop the channels usage and improving wireless resources, as well as the routing of data in multi-hop and multi-path scenarios.

3.4.1 ARCHITECTURE OVERVIEW

To allow a SDN based VANET system, suggested architecture involves the following SDN components:

SDN controller

The central logical intelligence of the SDN based VANET system. The SDN controller is responsible to determine the whole performance of the network.

SDN wireless node

SDN controller is controlling the data plane elements. And in this case, they are the vehicles that receive control message from the SDN controller to perform actions.

SDN RSU

Stationary data plane elements, which are controlled by the SDN controller. They are the infrastructure RSUs that are deployed along road segments.

The architecture expands SDN to work in mobile wireless VANET scenarios. In the proposal architecture various wireless technologies for controlling and forwarding planes are chosen as anticipated in future VANET systems: LTE/Wimax for control plane, and high bandwidth wireless connection i.e., Wi-Fi for data plane. The practical reason is that in VANETs not all nodes are easily accessible from the Infrastructure via RSUs. Figure 3.12 depicts the Software-Defined VANET communication between the components.

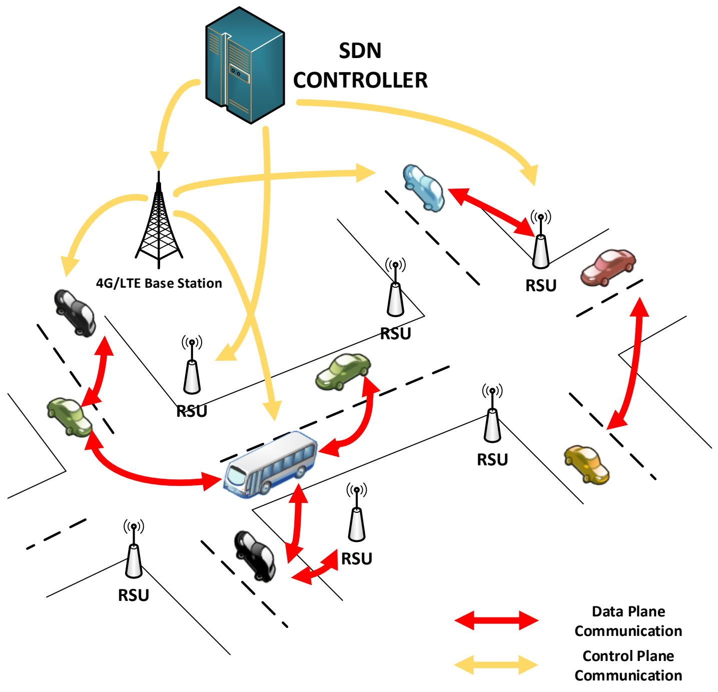


Figure 3.12 Software-Defined VANET communication

Figure 3.13 illustrates the components that are inside of the SDN wireless node. It includes all features of an OpenFlow-enabled switch in traditional OpenFlow networks, and additional intelligence to allow the performance of different modes in VANET environments. The SDN module is the aggregation of packet processing and the interface that adopts input from a separated control plane.

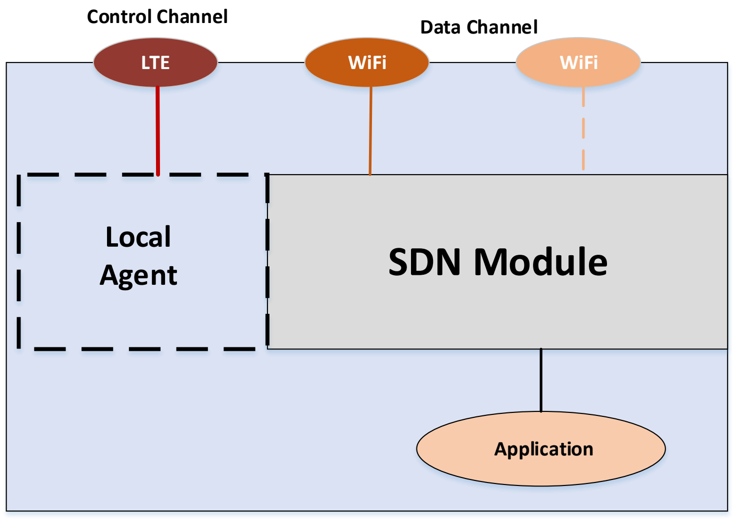


Figure 3.13 SDN wireless node internals

One specific trait of Ad hoc networks is that the nodes act simultaneously as Host and Routers. Therefore an SDN wireless node represents an SDN data plane forwarding element and an end-point for data. Traffic from any wireless node will run via its own SDN module before being sent, which allows the SDN controller to define the access of user traffic into the network.

3.4.2 OPERATION OVERVIEW

There are several ways in which Software-Defined VANET can work founded on the degree of control of the SDN controller, nevertheless the main idea of SDN which is decoupling control and date plane. A classification of this architecture is presented into three operational modes:

Central Control Mode

This is the operation mode, where all the operations of underlying SDN wireless nodes and RSUs are controlled by SDN controller. All the actions that the SDN data element implements are specifically determined by the SDN controller. The SDN controller will push down all the flow rules on how traffic should be manage as shown in Figure 3.14.

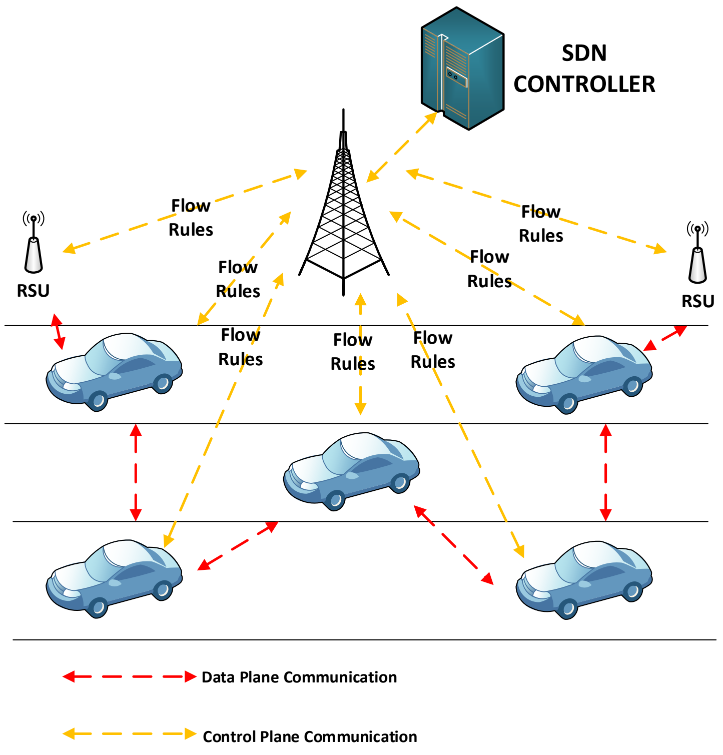


Figure 3.14 Central Control Mode

Distributed Control Mode

During packet delivery the underlying SDN wireless nodes and RSUs work without any directions from the SDN controller, in this operation mode. In essence this mode is very common to the original self-organizing distributed network, which do not have any SDN functions, expect that the local agent on each SDN wireless node controls the actions of each individual node, as illustrates in Figure 3.15.

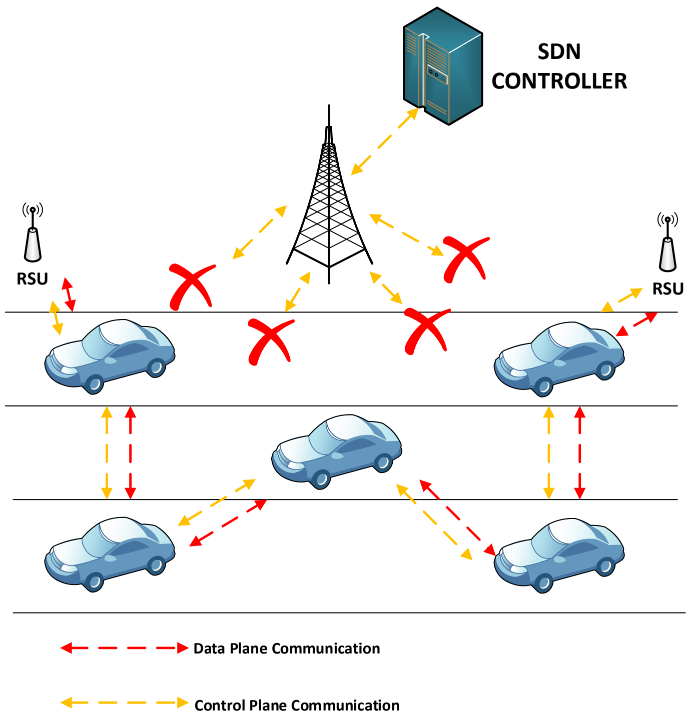


Figure 3.15 Distributed Control Mode

Hybrid Control Mode

All the operational modes of a system where the SDN controller influence anywhere between full and zero is included in this mode. Figure 3.16 depicts an example, where the controller does not have the total control, but on the other hand can assign control of packet processing details to local agents. For this reason the control traffic is transmitted among all SDN elements. For instance, instead of sending all flow rules, the controller will send policy rules instead, which determine the main behaviour, while the wireless nodes and RSUs use local intelligence for packet transmission and flow level processing.

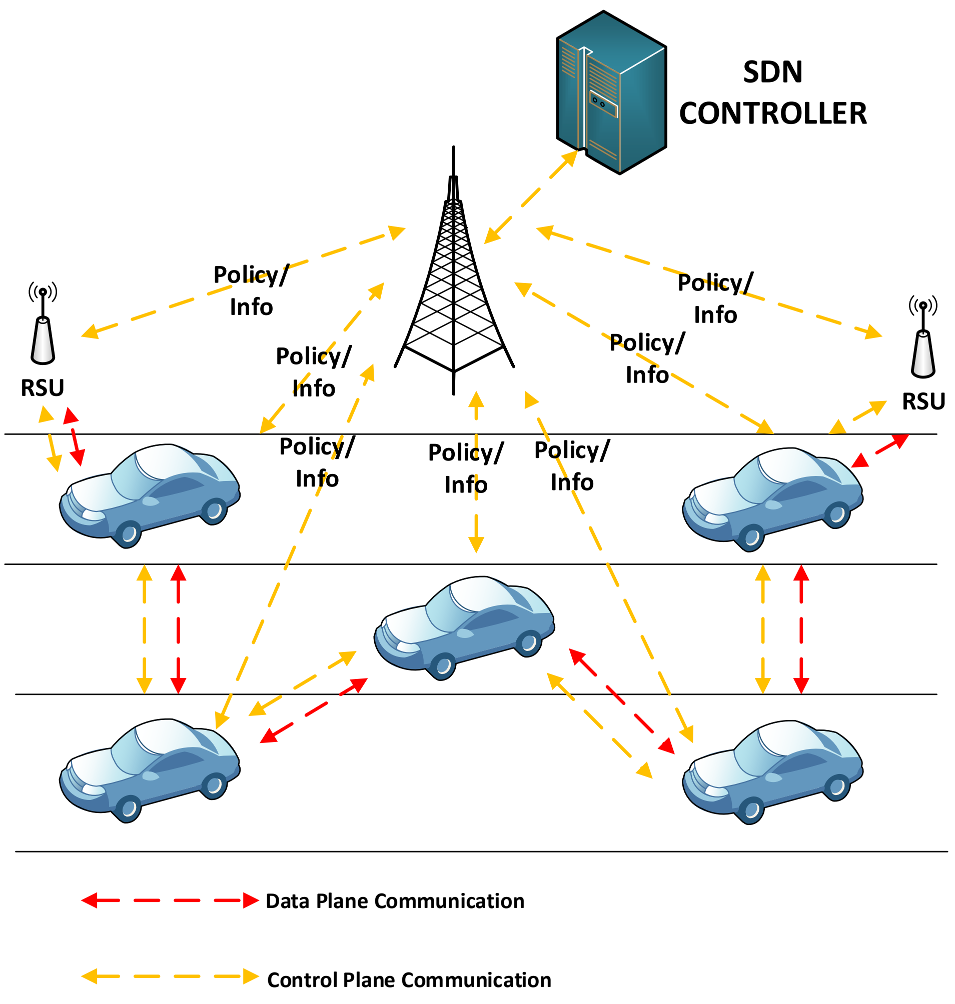


Figure 3.16 Hybrid Control Mode

3.4.3 SOFTWARE-DEFINED VANET BENEFITS

The implementation of SDN can bring several benefits for VANET networks and they can be divided into three different directions, namely Path selection, Frequency/channel selection and Power selection. Next, we are presenting the nature of each of them.

Path Selection

In first section of this work we explained that the routing decision in SDN networks are more informed due to the current network status that the controller can obtain instantaneously. In the case of VANET network data traffic can become unbalanced. This issue can occurs because of two reasons. Firstly, because the shortest path routing leads traffic focusing on some selected nodes and secondly because the application is video dominant and that is resulting in occupation of big bandwidth on the path. SDN is able to detect this problems and the controller can choose another traffic routing process in order to improve network utility and to reduce congestion.

Frequency/Channel selection

In SDN usually the wireless node has several available wireless interfaces or configurable radios (cognitive radios) [73, 74]. For SDN-based VANET allows improvements in the coordination of using channel/frequency. The dynamic nature of the controller allows the network to make a decisions for which time what type of traffic will use which radio interface or frequency. For example, this feature of SDN can be used in VANET emergency services because it allows to reserve channels for this type of traffic.

Power selection

In SDN-based VANET the system can make logical choice about transmission range due to the instant current status of the network. It can change the power of wireless interfaces dynamically. For instance, the controller can collect neighbour data from the wireless nodes to estimate if that node density is too sparse. In this case it can determine increments in the power of all nodes in order to achieve good packet delivery and to reduce interference.

3.5 SOFTWARE-DEFINED VANET SERVICES

Implementation of SDN technologies in VANET networks can lead to emergence of new types of services or to enhance these that already exist. In [8] the authors summarize software-defined VANET applications. They divide the enhancements in three directions, namely SDN Assisted VANET Safety Service, SDN-based On Demand VANET Surveillance Service and Wireless Network Virtualization Service. In this subsection we are exploring them.

SDN Assisted VANET Safety Service.

The major aim of using vehicle-to-vehicle communications is to improve road safety. In the previous subsections it is shown how a SDN VANETs can improve the services compared to conventional methods. The emergency and/or privileged traffic can take advantage of the reservation or limitation of specific frequencies using SDN. The difference in SDN that can enhance the usage of these channels is that the reservation is dynamically configurable. SDN controller can observe current traffic conditions and the needs of the applications and based on that information can assign or remove flows to the reserved channels. Additionally, using policies SDN can provide different level of services, by changing flow rules during an emergency time period. In that way, the emergency traffic takes priority over the remaining normal traffic.

SDN-based On Demand VANET Surveillance Service:

SDN approach can be used to enhance another service of traditional VANET network, namely Surveillance service for emergency/authority vehicles. The conventional way of providing this service contains request for the surveillance data sent by a requester (for example, police car), but in SDN based VANETs this request is done by the controller. It can just insert flow rules for the surveillance information and in this way it can reach the requesting nodes. In addition, when the same information is requested by multiple police cars, the SDN controller inserts flow rules and in that way the same copy of the data is sent to more than one destinations.

Wireless Network Virtualization Service

In the first section of that work we described how SDN can enhance network virtualization services. We already know that the purpose of it is to provide abstract logical networks over shared physical network resources and SDN has been used in data centres to achieve this aim. The same idea can be deployed for Software-Defined VANETs. The way that can be done is to force different flows choose different radios/interfaces using different frequencies. SDN can effectively bring different radio frequencies for each network to create virtual wireless networks with isolated traffic for each other. One approach can be the grouping of wireless nodes and RSUs. In that way each RSU can forwards traffic just from a selected group of nodes. Another more sophisticated mechanism can be to incorporate time slicing. This time slicing is performed by the SDN controller. It consists control on which network uses which radio interface or frequency for given time period. This allocation of network traffic is be done by programmable manner. Long Term Evolution (LTE) networks are using time slicing for Orthogonal Frequency Division Multiplexing (OFDM) spectrum allocation. OFDM can be used in SDN-based VANETs to support one virtual wireless network per time slot. Subsequently, if multiple radio interfaces are available, several virtual networks can be supported in the same time slot. For instance, ITS traffic is transmitted by frequency channel f_1 and MPEG DASH video uses frequency channel f_2. The video packet broadcast on channel f_2 is picked up by all neighbours assigned to f_2. SDN controller chose the nodes that will receive and forward the video packets. Furthermore, the SDN controller can apply filters on node inputs. In that way rejection of certain traffic class can be done. For example, that can help in restriction of propagation of video surveillance traffic to vehicles that are not police cars.

4 SECURITY CHALLENGES IN SDN AND VANET

4.1 SECURITY PROBLEMS SDN

In the past years one of the main concerns of governments and agencies worldwide are the cyber – attacks against government units, energy facilities, financial institutions and etc. This attacks are able to compromise and seriously damage a nation’s wide infrastructure which is a major issue. Normally the most common mean of performing those attacks is over the network, which could be the either the Internet or LAN. For example, although the attacker may has only a low capacity network connection at his premises it can launch large-scale attacks in networks having high capacity.

In SDN some of the main priority are the reliability and security due to the cyber-attack and growing trend of digital threats. Commercial adaption of SDN is still in its early phase although Google, Microsoft, Yahoo have been conducted a lot of experiments and research on it. A main concern today is the accessibility of Internet routers thanks to the widespread of clouds in terms of reliability. If SDN control platforms want to become an essential part of network applications reaching a high level of availability is crucial.

Some security problems and vulnerabilities in OpenFlow-based networks as well as various threat vectors in SDN architecture [75] have been already identified. There are some threat vectors specific directed to SDN like attacks on control plane communication and centralized controller. It is important to mention that the majority of the threat vectors do not depend of the technology or the protocol (OpenFlow, POF) because they are threats on conceptual and architectural layers of SDN itself.

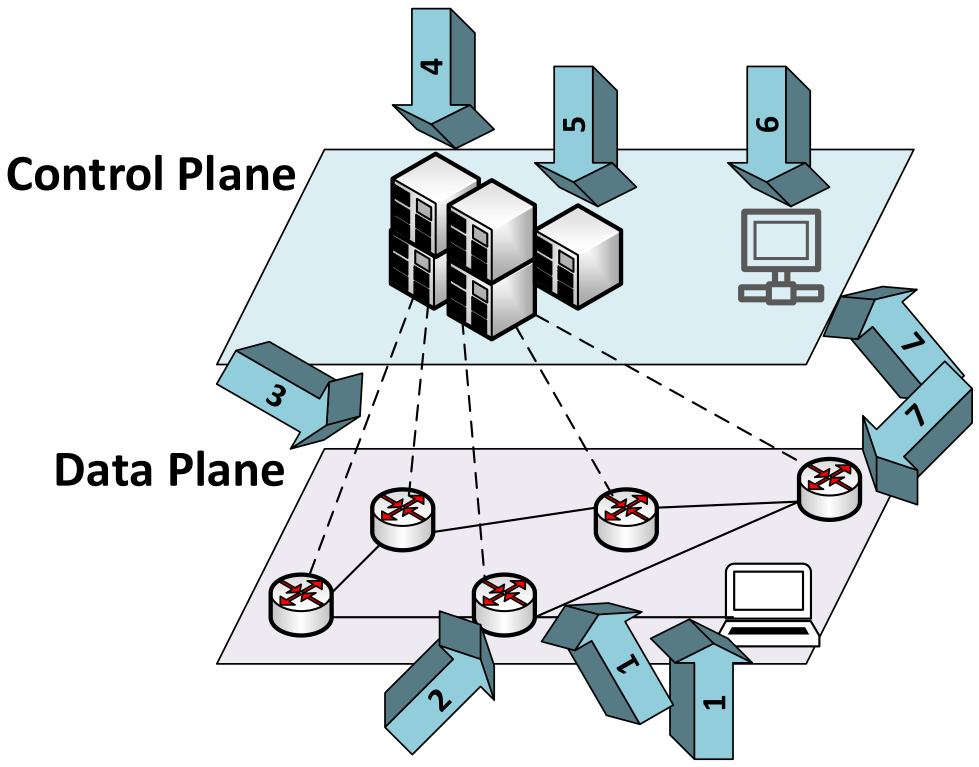


Figure 4.1 Main threat vectors of SDN

To attack forwarding devices and controllers as shown in Figure 4.1 and Table 4.1, there are at least first threat vector consists of forged or faked traffic flows in the data plane. By using the weaknesses of forwarding devices the second vector allow the attacker to cause chaos in the network. The most critical ones because network operation can be compromise are threat vectors three, four, and five. The attacker can take control of the network easily due to attacks on the control plane, controllers and applications. For example, damaged or malicious application on controller maybe be used to reprogram the whole network for any kind of malicious activities. The next one vector number six is associated with attacks on and weaknesses in administrative stations. The last threat vector depict the lack of trusted resources for forensics and remediation, which can compromise investigations and preclude fast and secure recovery modes for bringing the network back into a safe operation condition. As can be seen in Table 4.1, vectors 1, 2, 6 and 7 are non-specific to SDN and they were already introduced in traditional networks. Whereas threat vectors 3 to 5 are specific to SDN as they result from the separation between the control and data planes and subsequent inclusion of a new entity - the logically centralized controller.

Threat vector number	Is it specific to SDN networks?	Consequence in SDN
1	NO	Easily reachable for DDoS attacks
2	NO	Potential attack inflation
3	YES	Using logically centralized controller
4	YES	The entire network could be compromised from compromise controller
5	YES	Elaboration and implementation malefic apps on the controller
6	NO	Potential attack inflation
7	NO	Negative influence on fault diagnosis and quick recovery

Table 4.1 SDN Specific versus Nonspecific Treats

OpenFlow networks are subject to a variety of security and dependability problems such as spoofing, repudiation, tampering, information disclosure, denial of service, elevation of privileges, and the assumption that all applications are benign and will not affect SDN operation [76]. The lack of protection, isolation, access control, and stronger security recommendations are some of the reasons for these vulnerabilities.

4.1.1 OPENFLOW SECURITY ASSESSMENT

Number of different security problems are already identified in OpenFlow-enabled networks. Table 4.2 summarizes some attacks (based on [76]). For example, information detection can be obtain through side channel attack directed to flow rule setup process. Receiving information about the network operation is comparatively easy when reactive flow setup is in place. An attacker can easily conclude that the target network is a reactive SDN and proceed with a specialized attack by measuring the delay experienced by the first packet of a flow and the following. The attack is known as fingerprinting [77] - may be the first step to start a DoS attack designed to exhaust the network resources, for instance. Assuming its forwarding rule policies is harder if the SDN is proactive but still can happen [76]. An interesting fact is that all of the announced attacks and threats influences all versions (1.0 to 1.3.1) of the OpenFlow specification. It is important to note that some attacks, for example spoofing, are not specific to SDN, nevertheless they have a huge impact in these kind of networks. The attacker can take control of the whole network by spoofing the address of the network controller using a fake controller. Some smart attacks may continue only several second, i.e., just the time needed to install special rules on all forwarding devices for its malicious actions and therefore those kind of attack are very hard to identify.

Attack	Security Property	Examples
Denial of Service	Availability	Controller overhead from flow request
Spoofing	Authentication	Counterfeit ARP and IPv6 router ad, IP and MAC spoofing
Repudiation	Non-repudiation	Setup rules, amendment for fake source address
Tampering	Integrity	Modification which affects data plane, setup rules, counter forgery
Information disclosure	Confidentiality	Side channel attack to understand flow rule modifications
Elevation of privilege	Authorization	Controller take-over exploiting implementation drawbacks

Table 4.2 OpenFlow Network attacks

Another example is counter falsification, where an attacker can try to assume what are the installed flow rules and, thereafter, forge packets to artificially increase the counter. This kind of attack could be critical for load balancing systems and billing, for example. A load balancing algorithm may cannot take optimal decisions due to forged counters and a customer maybe be charged for more traffic than it actually used. Lack of security recommendation for developer are included in flow networks, the belief that TCP is enough since links are ‘‘physically secure’’ [78], the lack of TLS and access control support on most switch and controller implementations, enabling the creation of malicious TCP connections , for example, due to the fact that many switches by default have listener mode activated [78]. In conclusion, the high risk of Denial-of-service (DoS) posed to centralized controllers is important to mention, as well as the weaknesses in the controllers themselves, flooding attacks, bug in application, and the risk of resource depletion attacks [78]. By operating a single application such as a learning switch [78] an attacker can easily compromise control plane communication through DoS attacks and start an exhaustion of resources on the control platforms.

Another aspect that should be noted is that present controllers like Floodlight, Beacon, POX and OpenDaylight, have some drawbacks such as security and resiliency issues. To crash the current controller application problems (bugs) such as continuous allocation of memory space or abruptly exit of an application are enough. In terms of security, a simple malicious action in the data structure can affect and crash down the existing controller. There is still a long way to go in terms of dependability and security as these examples shows.

4.1.2 COUNTERMEASURES FOR OPENFLOW-BASED SDNS

To reduce the security threats in SDNs various countermeasures can be embedded. Several number of countermeasures which can be applied to various elements of an SDN/OpenFlow-enabled network are summarized in Table 4.3. In the current versions of OpenFlow specifications (version 1.3.1 and later) some of the measures such as event filtering, rate limiting, shorter timeouts, flow aggregation and rate limiting are already recommended. But most of them are not yet performed or supported in SDN deployments.

To mitigate or prevent attacks several approaches such as attack detection mechanisms, firewalls, access control and intrusion detection can be used and they can be applied in various devices like middle boxes, controllers, forwarding devices and etc. For instance, middle boxes can be a good choice for imposing security policies in an enterprise since they are more robust and high-performance devices. These method also decrease the potential overload which can occurs by applying these countermeasures directly on forwarding devices or controllers, but they can also increase the complexity of the network management.

Measure	Description
Attack detection	Applies mechanisms for finding various types of attacks.
Access control	Ensure authorization and authentication mechanisms on devices.
IPS and Firewall	Tools, which can fend from different types of attacks by filtering the traffic.
Event filtering	Permit or block certain types of events to be handled by specific devices.
Forensics support	For discover the originator of the problems by enabling reliable storage of traces of the network activities.
Flow aggregation	For fending DoS attacks and information disclosure by coarse-grained rules to match multiple flows.
Packet dropping	Devices have the right to reject packets due to security rules or ongoing system load.
Intrusion tolerance	Although intrusions allow the control platforms to support correct operations.
Shorter timeouts	Used for reducing the impact of an attack that deflects traffic.
Rate limiting	Support rate limit control to prevent DoS attacks on the control plane.

Table 4.3 Countramersures for Security Threats in OpenFlow Networks

To weaken various attacks such as DoS and information disclosure techniques like rate limiting, flow aggregations, packet dropping, shorter timeouts can be implemented on the controllers and forwarding devices. Packet dropping and rate limiting can be implemented to avoid DoS attacks on the control plane or to prevent ongoing attacks directly on the data plane by applying specific rules on the devices where the attacks is being originated. The attacker, with reduces timeouts would be compiled to steadily generate a number of forged packets to prevent timeout expiration and the attack can be easily detected.

Forensics and remediation encompass mechanisms such as secure logging, event correlation, and consistent reporting. The operators should be capable if something wrong happens with the network to safely found from where the threat occur and put the network to safety operation mode as fast as possible. Furthermore, to increase the robustness and security different techniques to tolerate intrusions and faults, such as reactive recovery, state machine replication, proactive–and diversity can be added to the controller. SDN controllers should be capable to stand against different types of attacks and events [79]. Replication is one of the most used traditional techniques to achieve high availability. Two examples of key techniques- proactive–reactive recovery and diversity which add value to the system for resting against various types of failures and attacks.

To address different threats and issues of SDN other countermeasures involve enhancing the security and dependability of controllers, protection, and isolation of applications trust management between controllers and forwarding devices, integrity checks of controllers and applications, forensics and remediation verification frameworks and resilient control planes [79]. Mandatory part of any controller should be the isolation and protection mechanisms. It is important applications to be separated from each other and from the controller. To avoid security problems from network applications different mechanisms like data access protection and security domains should be put in place.

Another crucial necessities is the implementation the trust between the controller and forwarding devices which should ensures that contaminated elements cannot damage the network without being detected. By spoofing the IP address of the controller an attacker can take control of the switches and make them to connect to its own controller. At the moment this is the case because most switches and controller only determine insecure TCP connection. To secure that safe code is being started once the system restarts additional integrity checks on controller and application software can help. Other specialized detection system should be developed for SDN, except the integrity checks.

Declarative languages to eliminate network protocol vulnerabilities are other methods for handling security threats in SDN which should be mention. They can determine structural constraints, semantic constraints, and safe access properties of OpenFlow messages. This kind of languages can help to locate and remove implementation susceptibility of southbound specifications.

Basic security properties such as authentication and access control are already start to shown. A certificate-based authorization, authentication, and accounting (AAA) architecture called C-BAS for enhancing the security control on SDN facilities. Solution which related to C-BAS can be made highly dependable and secure through hybrid system architectures combing different mechanisms and technologies form security, distributed systems, and fault and intrusion tolerance.

4.2 SECURITY PROBLEMS IN VANET

Until now security got less attention among all the challenges of the VANET. It is extremely important to be sure that VANET packets are not entered or changed by the attacker since they include life critical information. Similarly the responsibility of drivers should also be settled that they report the traffic environment correctly and within time. General communication networks do not experience those kind of security problems. The performance is difficult and different from other network security due to mobility, geographic relevancy and the network size.

The challenges of security must be take into consideration during the design of VANET architecture, cryptographic algorithm, security protocols etc. Some security challenges are introduces in the list below [80]:

• Real time Constraint: When it comes to safety messages which should be delivered with 100ms transmission delay VANET is time critical. Fast cryptographic algorithm must be used to reach real time constraint. Entity authentication and message should be done in time.

• Data Consistency Liability: Actually even node which is authenticated can perform malicious actions that can damage the network. To avoid this type of incoherence adequate mechanisms should be projected. They may be avoid by interrelation among the received data from different node on particular information.

• Low tolerance for error: On the basis of probability some protocols were designed. The actions in VANET are execute in very short time since they uses life critical information. A crash of the system may occur even with small error in the probabilistic algorithm.

• Key Distribution: In VANET all the performed security mechanisms dependents on keys. Each message is encrypted and need to decrypt at receiver end either with same key or different key. A major challenge in designing a security protocols is the distribution of keys among vehicles.

• Incentives: Manufactures are try to develop applications that consumer likes most. For instance, a few number of customers will be agree with vehicle witch announce any traffic rule violation automatically. Therefore successful implementation of vehicular network will demand incentives for vehicle manufacturers, government and the consumers is a challenge to implement security in VANET.

• High Mobility: Due to the high mobility of VANET nodes demands less execution time of security protocols for same throughput that wired network produces although the energy supply and computational capability are the same in both. To decrease the execution time appropriate methods should be used when designing security protocols.

• Low complexity security algorithms: Current security protocols such as SSL/TLS, DTLS, WTLS, usually uses RSA based public key cryptography. RSA algorithm uses the integer factorization on large prime no. which is NP-Hard. Therefore decryption of the message that used RSA algorithm becomes very time consuming and complex. AES can be used for bulk data encryption AES.

• Transport protocol choice: DTLS works over connectionless transport layer and should be preferred over TLS for securing the transaction over IP. Something that should be avoided as it demands too many messages to set up is IPSec which secures IP traffic. Nevertheless when the vehicles are not in motion TLS and IPsec can be used.

4.2.1 SECURITY REQUIREMENTS IN VANET

1. Authentication

Every message in Vehicular Communication must be authenticated, to ensure for its origin and to control level authorization of the vehicles, to this each vehicle will assign every message with their private key along with its certificate, at the receiver side, the receiver will receive the message and check for the key and certificate once this is done, the receiver verifies the message [81], [82]. Entering each message with this, lead to overload, by using the ECC (Elliptic Curve Cryptography) approach this overload can be reduce, the efficient public key cryptosystem, or we can sign the key just for the critical messages only.

2. Availability

Vehicular network must be available all the time, many applications will demand real time network and they will need quick response from sensor networks or Ad Hoc Network, and even a slight delay of a few seconds for some of the applications can make the message worthless and the results will be destructive [82]. Attempting to meet real-time requires makes the system attackable to the DoS attack. For example, in some messages a delay in milliseconds makes the message worthless; but when the application layer is trustless the problem is even bigger, because the potential way to bring back with unreliable transmission is to accumulate partial messages and in hopes to be accomplished in next transmission.

3. Non-repudiation

Even the attack is already happens non-repudiation will facilitate the capability to discover the attackers [82]. In TPD will be stored any information linked to the car: speed, time, any violation and any government in case of need will have the right to access this information.

4. Privacy

The preservation of the drivers information such as trip path, speed, real identity etc. away from unauthorized observers. The privacy could be reached by using provisional keys, and each key could be used once and outflow after usage [81]. TDP will kept all the keys, and when vehicles again makes an official check-up the keys will be reloaded again [82]. Electronic License Plate (ELP) is used to keep the driver real identity, and every new vehicle will get installed this license from the factory, it will ensure an identification number for the vehicle, and so it can be identify anywhere with the Radio-Frequency Identification (RFID) technology to hold the ELP. For example, if the policy or any other authorized institutions want to recover identity of specific vehicles ELP they could ask for permission and obtain the information they need.

5. Real-time constraints

Sometimes results will be devastating since the vehicles move in high speed and they will need a real time response quickly in many situations [82]. On the basis of an extension to the IEEE 802.11 technology, present ideas for vehicular networks rely on the emerging standard for DSRC.

6. Integrity

Integrity for all messages should be defended to stop the attackers from altering them, and the connections between the messages to be to be trusted [81].

7. Confidentiality

To prevent outside observers to have driver information the privacy of each driver must be secured and messages should be encrypted. [81]

4.2.2 DIFFERENT TYPE OF ATTACKS IN VANET

The expansion of internet and its extension to the vehicles has made our lives more comfortable, but this is also causes more abuse of technology and increase the number of cyber-crimes. Particularly in the cases of security and privacy VANET has still meet a lot of challenges, nevertheless all the advantages.

Sources of attacks and malicious activities in VANET [83]:

The degree of seriousness of attacks launched by the attackers can fluctuate based on the motif of the attack and the possible impact on the victim:

- Jamming: To avoid VANET communication between vehicles in a certain reception scope interfering transmissions are knowingly generated by the jammer.

- Forgery: Adulteration in VANET architecture compromises the validity, correctness, and timely receipt of transmitted data. One of the main vulnerability is the transfer of false warnings and those being received upon by all vehicles give a rise to chaos in the driving zone.

- Impersonation: When vehicle owner knowingly and hideously take the control and identity of another vehicle to its own or vice-versa is known as impersonation. It also includes message alteration, fake message fabrication and message replay.

- Privacy: Privacy is a serious problem in VANET, as the illegal surveillance of driver's personal data, could disturb their privacy. Due to the periodic and frequent nature of vehicular traffic driver privacy attacks are a severe vulnerability in VANET. The personal data of the drivers can be reinstated by the help of illegal in-transit traffic tampering of safety and traffic related messages, which are sent by the driver.

Considering the fact that the initial idea of VANET aimed to include mobile connectivity between the vehicles to give a rise of data transferring while traveling, in VANET cars have been victims to viruses, phishing, forged messages, identity thefts and many other threats. Security in vehicular environment is of paramount importance since incorrect massage could directly affect human life, particularly in the light of the public acceptance of the technology.

Since in a particular DSRC range the vehicular network is open and reachable from everywhere, it is estimated to be an easy target to users who has malicious intentions.

Attacks against routing

Due to the fact that the routing is the base of VANET communication and thus routing is also the most attackable part of VANET, sensitive to malicious operations and attacks [84]. Infected nodes in VANET take advantages of the routing algorithms that works together to start routing attacks, such as BH and rushing attacks. Into two broadly categories attacks against routing in VANET are classified:

Impersonating: This attack includes taking on the credentials of another vehicle to spoof route messages. It is also contains fake route metrics advertisement to confuse the topology, sending route message with wrong sequence numbers to repress/delay other messages that are consistent. Flooding the route find unreasonably with DoS, amending a RREQ messages to set false routes, producing bogus route error messages to interrupt a working route or suppressing a valid route error to prevent other vehicles to reach information for further attacks that can be started by the impersonator.

Application Attack: Applications linked to safety and comfort are the major directions of developing new VANET application. Attacker target these application linked massages to take advantages of them for their own benefits, to the detriment of other users. Attacker amendments the content of the real messages and send false, partial, modified or fake messages to other vehicles entailing to serious traffic congestions or even accidents.

One of the most frequent types of attacks application is the bogus information attack, where an attacker inserts false information into the network and these wrong/ fabricated messages have a direct impact over the conduct of cars on the road.

Another catastrophic attack in this category is the modification/alteration of warning messages, which contain the degree of authenticity of a message in the VANET architecture.

Timing Attack: The main goal of this attack is to make the initial massage delay, by adding extra time slot to the original message. By adding this additional time they make the messages useless nevertheless they have unchanged content. Even a small delay in the transmission of messages in VANET can affect these time depending applications.

Social Attack: Class of attack in which by forwarding immoral messages the attacker change/aggravate the behaviour of legitimate vehicles.

Monitoring Attack: Monitoring and tracking of the vehicles, illegally listening to the communication between V2V and V2I and abusing any confidential information is the goal of this attack. Table 4.4 shows various types of security attacks with attacker types and respective security attributes.

Attack Name	Attacker type	Security attributes /requirements	PHY access?	Communication types
Bogus Info	Insider	Data Integrity/Authentication	-	V2V
DoS	Malicious, active, insider, network attack	Availability	Yes/-	V2V; V2I
Masquerading	Active, insider	Authentication	Yes	V2V
Black Hole (BH)	Passive, outsider	Availability	Yes	V2V
Malware	Malicious, insider	Availability	-	V2V; V2I
Spamming	Malicious, insider	Availability	Yes	V2V
Timing Attack	Malicious, insider	Data Integrity	-	V2V; V2I
GPS Spoofing	Outsider	Authentication	-	V2V
Man-in-the-middle	Insider, monitoring attack	Data Integrity, confidentiality, privacy	Yes	V2V
Sybil	Insider, network attack	Authentication, privacy	Yes	V2V
Wormhole/tunneling	Outsider, malicious, monitoring attack	Authentication, confidentiality	Yes/-	V2V
Illusion attack	Insider, outsider	Authentication, data integrity	Yes	V2V; V2I
Impersonation attack	Insider	Privacy, confidentiality	Yes	V2V
Social Attack	Insider, e.g. “you re idiot”	Data integrity, trust	Yes/-	V2V
Monitoring attack	Monitoring the road activity	Privacy, authenticity	Yes/-	V2V; V2I

Table 4.4 Different types of security attacks in VANET with attacker types and respective security attributes

4.3 DOS AND DDOS ATTACKS – DEFINITION

Definition

A Denial of Service (DoS) attack can be defined as an attempt made by a malicious user to compromise the regular functioning of the network. If this attempt comes from a group of hosts, instead of only one host, we are talking about Distributed Denial of Service (DDoS). This group of hosts are coordinated by certain malicious user. To launching DDoS attacks the attacker uses botnets - large clusters of connected devices (e.g., cellphones, PCs or routers). These devices are infected with malware, this allows attacker to have a remote control on them. In this way, each infected device will send a huge amount of packets to the victim`s part, in order to exhaust its resources and make the network unavailable to the legitimate users. The attacker has the potential to send massive volume of packets to the target with spoofed source IP addresses. Later In this section we are explaining in details the nature of the botnet.

Denial of Service Attack Types

We can divide the DoS attacks into two main categories - application layer attacks and Network layer attacks. [85]

Application layer attack - it works on layer 7 of the OSI model and can be either DoS or DDoS attack. This vulnerable attempts aim to overload a server using a huge number of requests. This requests require resource-intensive handling and processing. In this category we can include other attack vectors such as HTTP floods, slow attacks (e.g., Slowloris or RUDY) and Domain Name System (DNS) query flood attacks. Usually the size of these attacks are measured by requests per second (RPS). To cripple a mid-sized website it is needed between 50 and 100 request per second.

Network layer attack – it works on layer 3–4 of the OSI model and in the general case is DDoS attack. These attacks aim to flood the “pipelines” connecting the network. Here, in this category, we can include type of attacks such as User Datagram Protocol (UDP) flood, SYN flood, Network Time Protocol (NTP) amplification and DNS amplification attacks. Later in this subsection we examine in details UDP and TCP flood, in order to use this type of attacks in our proposal solution. Usually this kind of attacks are used to prevent access to servers and they leads to tough operational damages. Some of them are account suspension and massive overage charges. Normally the size of these attacks are measured by gigabits per second (Gbps) or packets per second (PPS), because usually the traffic is really high. In practice, between 20 and 40 Gbps are enough to completely shut down most network infrastructures, but the largest attacks can reach 200 Gbps.

In [85] the authors are made classification based on different aspects of the attack. Based on:

- The degree of automation - manual, semi- automatic and automatic DDoS attacks.

- The exploited deficiencies - the different weaknesses exploited to deny the service of the compromised part to its clients.

- Source address validity - in case of IP spoofing source address is used or not.

- Attack rate dynamics – if the participating agent transmits at constant or variable rate.

- The persistence of agent set **-** whether the agent set is persistent (and thus can be traced) or not.

- The victim type - if the victim is an application, a host, a resource, the network or the infrastructure.

- The impact on the victim - disruptive and degrading attacks.

It is obvious that several types of attacks that belong to the above described classes can work in combination in order to set up a more vulnerable threat. A typical solution to block DDoS attacks are using of firewalls, but this action has to work together with Intrusion Detection Systems (IDS) to collect traffic based information and send it to an analysing system to detect malicious users.

The focus of our work is detecting a flood attack in SDN based VANET network environment. That’s why here we are going to examine TCP and UDP flooding attacks. We are going to use one of these attacks, but before choosing which type to use, we are exploring them in the next two subsections.

SYN Flood

Synchronization (SYN) flooding is type of DDoS attack. It is also called Half-open-attack [86] and it is a transport layer attack. It represents one of the most common type of network vulnerabilities. [87]

TCP connection is established by exploiting the Three-Way Handshake methodology. It can be seen on Figure 4.2. When the TCP SYN flood attack occurs the victim node enters a LISTEN state, because it receives a connection request using a SYN segment from a remote user. To enter this state the victim node does not need to receive the IP address. If a malicious node request a connection using a spoofed IP source address, the next SYN ACK reply from the victim`s side will be discarded by the network. In the general case these spoofed IP addresses belongs to a hosts that are unreachable or that have an anomalous response to the ACK segment from the victim. When the victim enters in a LISTEN state, it keeps a half open connection state while waiting for ACK response to the SYN ACK reply it sent before. The victim`s node waits for a certain amount of time and then stops. If the buffer on that side allows to store multiple packets, this technique will exhaust its available resources. If a large number of malicious coordinated users send fake TCP SYN packets, the destination will occupy a lot of resources and that will result in a Denial of Service to legitimate traffic. TCP SYN attack can be seen on Figure 4.3.

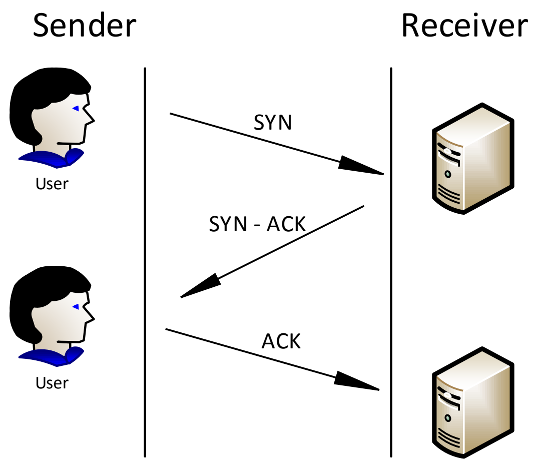


Figure 4.2 TCP Three-handshake procedure

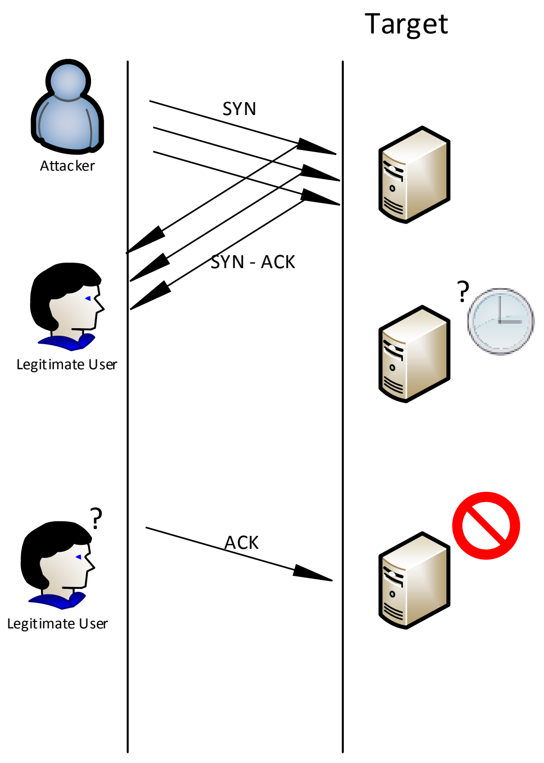


Figure 4.3 TCP SYN attack

UDP flood

Before looking into the UDP based DDoS attacks we are explaining the basic characteristics of this protocol. UDP is a datagram protocol which offers a minimal transport service. UDP does not provide reliability, datagram ordering and data integrity. Since UDP traffic does not support a three-way handshake like TCP, it runs with lower overhead and is designed for traffic that does not require to be checked and rechecked, such as Voice over Internet Protocol (VoIP).

The UDP packet (Figure 4.4) has a header and a payload. The header has only 4 fields, each of which is 2 bytes (16 bits) – Source port, Destination port, Checksum and Length. Some of these fields are optional, namely Source Port and Checksum. In IPv6 only the source port is optional.

Source port number identifies the port of the sender (if not used, this field is equal to zero). The port number is probable to be an ephemeral port number, if the source host is the client, but if the source host is the server, the port number is a well-known port number. Destination port number is a field which identifies the port of the receiver and it is required. Length is a field in the UDP header that specifies the length (in bytes) of the UDP header and data. The minimum length is 8 bytes (that is the length of the header).

16 bits	16 bit
Source Port	Destination Port
Length	Checksum
Payload	

Figure 4.4 UDP packet structure

In [88] the authors claim that during normal operation, the packet rate of traffic going to an address is proportion to the packet rate of traffic going from that address. This applies to TCP by protocol specification. Even though the UDP protocol specification does not require ACK message at the application layer, it is implemented a mechanism that ensure other party is still alive.

UDP flood is a type of DDoS attack associating with overwhelming of random ports on the targeted host with IP packets containing UDP datagrams. The fact that UDP is a connectionless networking protocol can make UDP more vulnerable to abuse. Usually every packet has the same Payload. The source IP address may be spoofed, but that is not mandatory. The research shows that if the address is not spoofed or not randomized the chance of more effective attack is higher. In case of a flooding attack it can observed one-way flood packets.

To initiate a host-based flooding attack, a UDP packet to a random port on the victim system are send by an attacker. As a response, the compromised system determines what application is waiting on the destination port. An Internet Control Message Protocol (ICMP) packet is transmitted to the forged source address, saying “destination unreachable”, if the destination does not find waiting application on the port. The system will go down if the amount of UDP packets on that port is large. ICMP flooding (known also as a ping-of-death attack) occurs when big amount of ICMP packets are transmitted to a server without waiting for replies from it. This can exhaust the resources of the server after a while and it becomes unable to accept more packets.

DrDoS attack

A DrDOS (Distributed Reflection Denial of Service) is another special type of DDoS attack. The main idea is to call a high number of servers (for example DNS – name server, NTP –

time server, or Quake, CoD – online videogame protocols…) by using a UDP [89]. Owning to the UDP, a third person’s IP address can be used to make the packets bounce and hide the initiator of the attack. The idea is to send, for example to a game server, a request for the list of current games. This request is made from the target’s IP, by changing his source IP address and it will take a few octets and the answer can take several hundreds of Kilo octets, sometimes even Mega octets. The attacker asks this gaming list, from IP address of the target, after that “invests” a few octets into hundreds of game servers. As a result the target host receives big waves of packets and bandwidth from all those servers. The efficiency of this attack depends on the multiplication coefficient between the size of the minimal request and the size of the answer. The higher it is, the higher impact of DrDoS.

It is also possible to make this type of amplification with a TCP DrDoS. As we already know the TCP protocol, makes sure before transmitting that the target is ready to receive the information (three way handshake). In addition it checks after the transmission that the whole data has been received (ACK reply). The procedure can be tried 5 times in a row. In case of TCP session opening (SYN request) the spoofed receiver will receive 5 tries for a session opening, that is to say a 10 coefficient.

4.4 BOTNETS

Botnet (Robot Network) is one of the most serious network security threats which home user, organizations and government can meet. A botnet is a group of many infected machines communicating with each other, called zombies, managed by a malicious entity called the botmaster. In other words, bots represent software programs that run on a host device permitting the botmaster distant control of the host actions. Botnets use command and control channels (C&C) to communicate with each other. C&C channels can use different communication protocols and can work over wide range of logical network topologies. Control and channel architecture determines the manner in which bots are controlled it can be HTTP, DNS or P2P-based [90].

Botnets are used to perform a cybercrimes, for example, stealing personal data, sending spamming emails or launching denial-of-service attacks. They are commonly named after malicious kits used in their establishment. Nevertheless, not all of the kits are detectable as botnet herders operate in anonymity. Frequently used DDoS botnets are [91]:

- Nitol / IMDDOS / Avzhan / ChinaZ

This is a periodically transformed DDoS botnet family, works mostly in China. Once is installed, Its malware commonly attach to the botnet`s C&C; server using a TCP socket and after that sends information of the effectiveness from the victim`s device.

- MrBlack

This malicious software aim to compromise the Linux platform, but is also applicable for different platforms and architectures. It is also known as Trojan.Linux.Spike. It sends system information by contacting a remote server. Furthermore, by receiving a control commands it accomplished various type of DDoS attacks against a certain target, download a file and implement it, and then terminate a process.

- Cyclone

DDoS malware which is develop in the U.S.A. The command and control specifications are blurred and it is IRC-based. It is known to eliminating off other bots on contaminated host. Attacks comprise plurality HTTP floods, Apache remote memory exhaustion (ARME) and SlowLoris.

- Pushdo / Cutwail

This is a botnet mainly focused at sending spam e-mails. In most cases the bot represent a computers that are infected running Microsoft Windows in the form of Trojan component called Pushdo. There is a statement from 2015, which describes how Pushdo botnet affect the computer user in more than 50 different countries.

DDoS attack using stationary botnets

The fast development of botnet technologies allows the generation of different types of DDoS attacks. In the past years the most complex ones have been launched using botnet technology. There are several reasons why most of the attackers choose to use this technology: given the similarity with the normal traffic they are difficult to be catch in real time; the identification of the real attacker is very complex; formation of powerful flooding attack due to the large number of zombies included in the network and bypassing security mechanisms by using protocols. Figure 4.5 illustrates simple example of DDoS attack which is launched by the use of botnet. There is a short description of the botnet parts below.

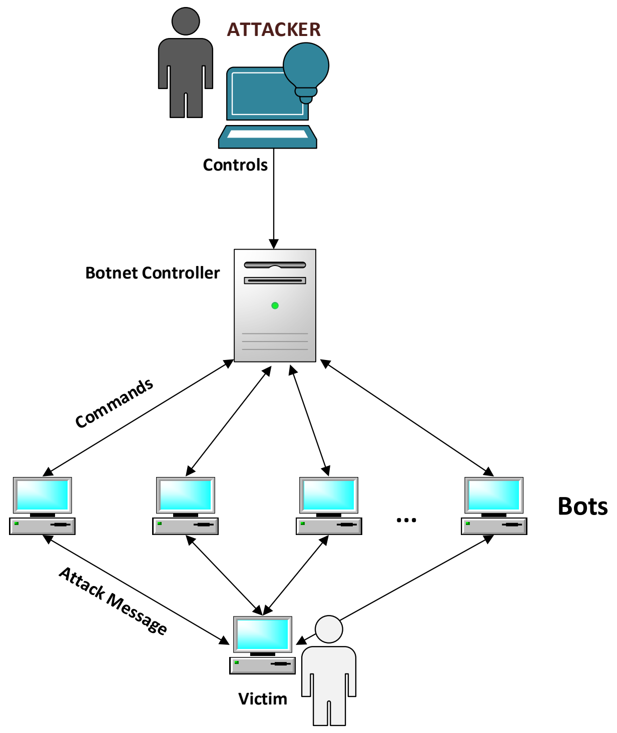


Figure 4.5 Botnet Attack

- Attacker - to launch an attack the attacker configures the bots. The first thing is to access the machine and set a malicious code and earn control of the machine once it connects the control server.
- Botnet controller - it can send and receive communication commands from the connected parties, because it operates like a command and control server.
- Compromised host - operates as a bot in a botnet after the malicious code have been installed in the machine.
- Victim - in that network environment we can categories the host that receive a huge number of attack packets as a victim(s).

Mobile Bonet

The vast and fast development of mobile device had leads to evolution of mobile botnets. They represent a set of threatened smartphones, which are remotely managed by a botmaster through C&C channel. Mobile botnets attract attention of the attackers to use them as platform for starting DDoS due to the fact that mobile devices can communicate with Internet services via different techniques like Universal Mobile Telecommunication System (UMTS), Evolution Data Optimized or Enhanced Voice Data Only (EVDO), General Packet Radio Service (GPRS), High Speed Downlink Packet Access (HSDPA), Enhanced Data Rates for GSM Evolution (EDGE). The limited battery power, non-fixed IP address and limitation of the network are reasons why most of the intruders use the mobile platform during the first steps of starting a DDoS attack. Figure 4.6 illustrates a mobile botnet architecture.

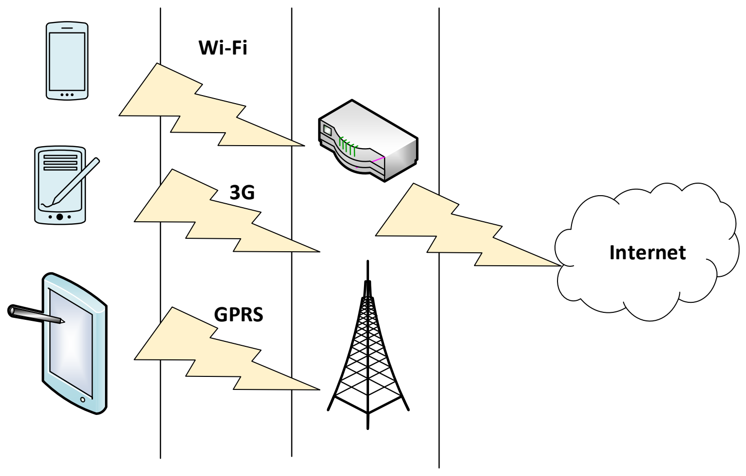


Figure 4.6 Mobile botnet architecture

The specific characteristics of mobile environment represent some challenges to malicious software and mobile botnet due to the fact that they are normally less secure. The botmaster is liable for managing the channels of affected nodes in a mobile botnet. The botnet could not be capable to operate if we can block the botmaster channel. In the general case a mobile botnet use three types of C&C mechanism- Internet-based or IP-based, GSM –based and Local Wireless C&C [90].

Transferring commands from the botmaster to the mobile bot is typically liability to C&C channel. During mobile attack the design of this channel is essential and should be done carefully. Normally, throughout the communication a mobile botnet use four different channels [90]: SMS C&C channel, Bluetooth C&C Channel, HTTP C&C Channel and Hybrid C&C Channel.

4.5 DDOS ATTAKS IN SDN NETWORKS

It is well known fact that SDN can bring a lot of benefits because of decoupling the control from the data plane. But still there is a vulnerable relation between SDN and DDoS attacks. SDN itself may be a target of DDoS attacks. Network capabilities, such as global view of the network, dynamic updating of forwarding rules and so on, can facilitate DDoS attacks detection, but the separation of the control plane from the data plane leads to emerging new types of attacks. [92] For instance, an attacker can use the characteristics of SDN to launch DDoS attacks against the control, infrastructure and application layers of SDN.

The impact of DDoS attacks can grow greatly if the attackers take advantage of botnet to overwhelm their victim's network. We have discussed this earlier in the previous subsection. Smartphones, tablets and other mobile devices are already a significant launching platform for DDoS attacks. They are perfect platform for attackers to launch DDoS attacks, because of the increasing band width and processing power. They can take advantage of the lack of security of these mobile devices as well.

In the following table 4.5 we can observe some good features of SDN in defeating DDoS attacks [92]:

Good features of SDN	**Benefits for defending DDoS attacks**
Separation from the control from data plane	It is able to establish large scale attack and defence experiments easily.
A logical centralized controller and view of the network	It helps to build consistent security police
Programmability of the network by external applications	It supports a process of harvesting intelligence from existing IDSs and IPSs
Software based traffic analysis	It improves the capabilities of a switch using any software-based technique.
Dynamic updating of forwarding rules and flow abstraction	It helps to respond promptly

Table 4.5 Good features of SDN in defeating DDoS attacks

SDN separates the data from the control plane, and in that way it is possible to build easily large scale attack and defence experiments. In is not complicated to perform experimentation in real environment, because of the separation among virtual networks [93]. Continuous deployment of new concepts can be done through a smooth transition from an experimental to an operational phase.

Based on the information that can be collect through requesting the hosts and remote authentication dial in user service (RADIUS) is possible to dynamically isolate compromised hosts and recognize legitimate hosts [92].

SDN networks can take advantage of existing intrusion detection systems (IDSs) and intrusion prevention systems (IPSs) due to the programmability of the network [94].

Software-based traffic analysis can be done using all types of intelligent algorithms, databases, and any other software tools. This feature considerably allows emergence of innovation.

Dynamic updating of forwarding rules is very useful feature against DDoS attacks in SDN. New or updated security policy can be launched to the network as flow rules to block the attack traffic without much of delay. That can be performed based on the traffic analysis [92].

In the next part of that subsection we are describing some of the possible DDoS attacks on SDN. In [92] the authors had classified DDoS attacks considering the layers in the network. As we already know from the first section of this diploma thesis SDN is vertically divided into three functional layers — infrastructure, control and application layer**.** So we can assume that the potential DDoS attacks can be launched on these three layers of the architecture (Figure 4.7).

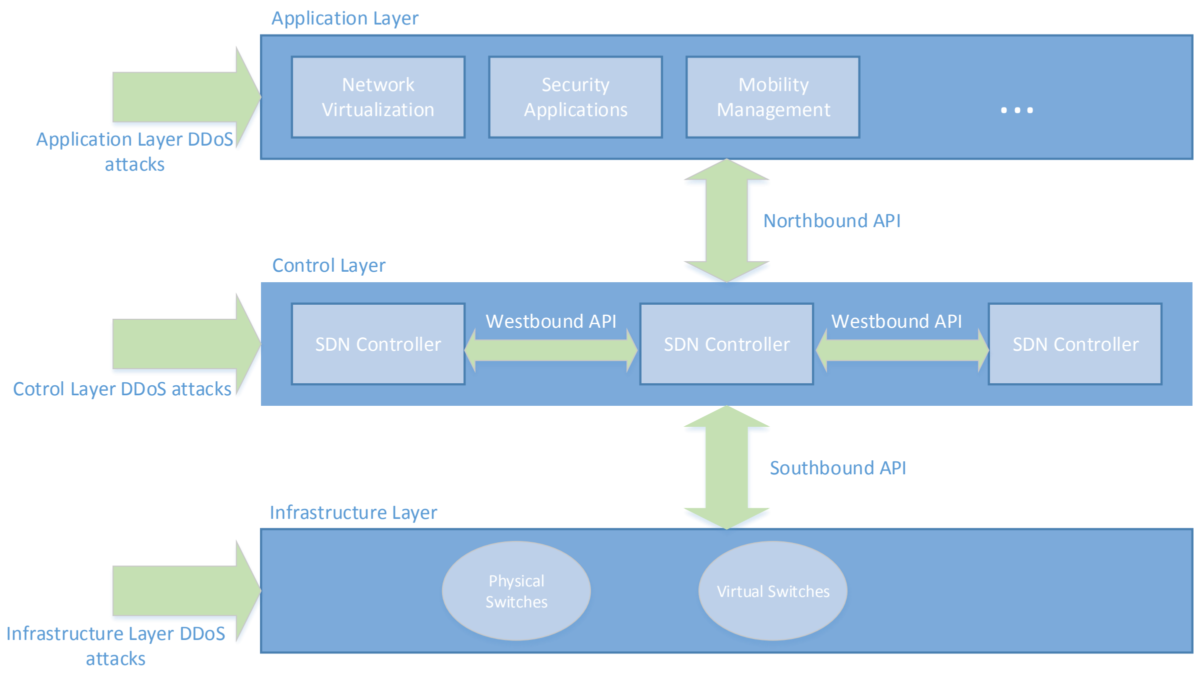


Figure 4.7 Potential DDoS attacks can be launched on these three layers of SDN's architecture

Here we are exploring how this attacks can be launched.

- Launching an attack on the application layer:

There are two approaches - attack applications or attack the northbound API. DDoS attacks targeting one application can affect other applications, because of that the isolation of applications in SDN is not well perfumed.

- Launching an attack on the control layer:

That is probably the most threatened part of the SDN architecture, because the controllers could be a single point of failure for the network. Unlike the application layer methods for launching an attack here the attacker can choose between more approaches: attacking the controller, the northbound API, the southbound API, the westbound API or the eastbound API. In the first section of that work we explained that in SDN networks the switches are belonging to the data plane and they request the control plane to obtain flow rules when the new packets arrived and the switches do not know how to handle them [95]. If a flow match doesn`t exists in the flow table, the controller receives either the complete packet header or a portion of it to resolve the query. If the complete packet is sent to the controller that would occupy high bandwidth. The attacker can take advantage of that functionality of the network and if he sends a large volume of traffic it will exhaust the resources of the controller.

In [92] and [96] authors exploited the limitation of the memory size - flow table size in Controller and Network Devices. The communication between the switches and the controller is through a secure channel and if this channel is disconnected, the whole network will lose its operating system. We can think about the controller as the brain of the network. If the switches can perform functions in normal operation mode - processing and forwarding, this loss can be not that harmful.

Another effect of DDoS attack that can be observed is the filling flow tables of switches. That is consequence of adding a new flow to that tables for every new incoming packet. The controller is responsible for that. That functionality of SDN can be used from the attacker to fill the switches with fake flows.

- Launching an attack on the infrastructure layer:

Here the attacker can choose between two methods. He can attack switches or attack the southbound API. For instance, because only the header of the packet is transmitted to the controller, the whole packet information must be stored in node memory until the flow table entry is returned. That’s why it would be easy for the attacker to launch a DDoS attack on the node by setting up a number of new and unknown flows [92].

Usually the DDoS attacks use forged source IP addresses or faked traffic, simple authentication mechanisms could work against that. But the attacker can easily use authenticated ports and source media access control (MAC) addresses to inject authorized, but forged, flows into the network, in case when he assumes the control of an application server which stores the details of the users [75].

Despite the fact OpenFlow provides support for encrypted transport layer security (TLS) communication and an authenticated exchange between the switches and the controller, it does not guarantee secure communications.

DDoS attack in Mobile SDN Environment

In [92] authors also consider the mobile DDoS attacks, but they claim that more research needs be done in the area of mobile DDoS attacks using SDN. The mobile devices and the applications that can potentially be installed on them can be used to initiate DDoS attacks. If we consider the growing trend of usage of mobile devices and cloud computing, we can conclude that the focus of DDoS attacks and defences will alternates from the traditional to the mobile cloud computing networks. Due to the fact mobile networks use super proxies, we can declare that the simple filter method based on source IP addresses cannot be adopt, because it will also block legitimate traffic.

5 USING ENTROPY FOR DDOS DETECTION

5.1 INTRODUCTION

In this chapter we are proposing a design of a model for DDoS attacks detection based on a statistical method using Entropy. Before looking into the entropy detection solution we examine what is entropy, its formulas and computation. After that we present a comparison of different existing methods. The topology of our method is discussed. Our scenario is based on SDN architecture for VANET. We will launch DDoS attack traffic and we will implement a DDoS module on the SDN controller. For the needs of anomaly detection in VANET control plane communication, we are going to rely on what is done in SDN research.

5.2 TYPES OF ANOMALY DETECTION TECHNIQUES

In Chapter 4 we did in depth analysis of DDoS attacks. Based on this, we can notice that the common aspect in all kinds of attack is pushing big amount of traffic into the network to exhaust its resources. Usually, in normal circumstances, a pattern in the network activity can be defined to specify the accepted rate of bandwidth consumption. To classify a traffic as abnormal it has to take into account a sudden increase in traffic, delay, CPU utilization, or sudden drop in performance of any of the network assets. In the general case, anomalies are related to the type of data in the network [97]. Understanding the nature of the transmitted information and its characteristics in the network is the first main step to detecting abnormality. Some of these characteristics can be packet header information, delay, packet size, protocol type, etc.

We can conclude that the characteristics of the network define the type of intrusion. If a network is vulnerable to a certain threat, then the efforts need to be focused on detection and mitigation of this kind of threat.

Every IP network has a certain bandwidth and processing power for transport the traffic. We can define pattern for each attribute of the network if we put this attributes to a statistical analysis. The reliability of the pattern depends on the time, longer time gives us more stable pattern, but this is true only if the traffic is steady all the time. Data collection, filtering and processing are well known methods for DDoS attack detection.

Statistical analyses (like Entropy and Chi-Square techniques) [98] and machine learning are two of the common methods of anomaly detection.

Entropy represents headers of the packets as independent data symbols with unique probability of occurrence. It is a common method for DDoS detection [99] [100] [101]. If we pick a window of some number, for example 10,000, and moving the window forward, we can find a pattern with probabilities for each kind of header If dramatically changes in the bins of each header occur (above the average bin limits) the system of anomalies will be alerted. This technique will be analysed further to examine its potential use in SDN. If we know the type of intrusion that can be expected and the type of packet header is better to use Chi-Square model. For example, if TCP SYN flood is the attack that can occur, then sampling bin of data and counting the number of TCP SYN headers will reveal a pattern of the average number of such headers. Any variance beyond the estimated limits is assumed abnormal.

5.3 USING ENTROPY TO MESURE THE RANDOMNESS

In this subsection we are going to explore what is entropy and we will focus on its formulation and computation. The main reason for choosing entropy for our solution is its ability to measure randomness in a network. The higher the randomness the higher is the entropy. The opposite is also true.

Entropy (Shannon-Wiener index) is an essential concept of information theory, which is a measure of the uncertainty or randomness associated with a random variable or in our case data coming over the network (destination IP address).

We are adopting the way of describing the entropy and its formulation of [79]. We assume we have a set of data W with n distinct elements and x is an event in the set (Equation 1). Then, the probability of x happening in W is shown in Equation 2:

$$W = \{x_1, x_2, x_3 \dots x_n\} \tag{1}$$

$$p_i = \frac{x_i}{n} \tag{2}$$

To measure the entropy (labelled by H), we have to calculate the probability of all elements in the set and sum that as shown in Equation 3:

$$H = -\sum_{i=1}^{n} p_i \log p_i \tag{3}$$

The entropy will have a maximum value if all elements have equal probabilities. Hence, if an element appears more than others, the entropy will be lower. The size of **W** is called the **window size.** In our case the data is packet header and it will be parted into equal sets that are called windows. In every window, each element and its appearance are counted. For example, if the window has 64 elements and, all elements appear only once, the entropy will be 1.80. If one element appears 10 times, the entropy will be 1.64. In this paper the entropy will be used for calculating the randomness in the SDN controller. In SDN, when passing packets to the controller, the limitation of available resources and the quick detection of attacks are key features of any detection scheme. In this paper, we will take into account the above limitation of the controller to apply entropy for DDoS detection.

5.4 MAIN COMPONENTS NEEDED TO DDOS DETECTION USING ENTROPY

To detect DDoS attack we have to define the following two essential components:

Window size - either based on a time period or is based on number of packets. Entropy is calculated within this window to measure uncertainty in the coming packets.

Threshold - The maximum and minimum values of entropy can be deterministic in the controller. If the calculated entropy passes a threshold or is below it, depending on the scheme, an attack is detected.

5.5 COMPARISON OF DIFFERENT DETECTION METHODS FOR DDoS ATTAKS USING ENTROPY

Here we are going to present different Entropy approaches used in literature for DDoS detection in order to explore what is done until this moment in SDN and VANET networks. Entropy has been used in different ways to detect DDoS attacks in the network but, to the best of our knowledge, it has not been used in SDN. Next three methods involve the components mentioned in the previous part of this chapter to DDoS detection using entropy, but they are not SDN or VANET-based cases. The first method is described in more details compared to the others because the parameters and the main approach that they used in [102] are also used in our solution and here we are going to introduce theoretically this parameters. In contrast, they use a one-sided test instead of threshold.

Oshima et al. [102] propose a "short-term statistics" detection method based on entropy. In this case short-term means that they are calculating entropy in small window size - 50 packets. They have tested different window sizes to find the best for optimal entropy measurement. Table 5.1 represents the results of these tests:

W	**H_N**	**H_A**	**$\lvert H_N - H_A \rvert$**	**S_N**	**S_A**	**z**
5	1.36	1.98	0.62	0.79	0.48	1.29
10	1.89	2.72	0.83	0.98	0.56	1.49
50	3.11	4.22	1.11	1.35	0.65	1.7
100	3.59	4.73	1.15	1.39	0.64	1.8
500	4.54	5.51	0.96	1.05	0.4	2.4
1000	4.88	5.67	0.79	0.78	0.32	4.48
5000	5.5	5.92	0.42	0.31	0.13	3.25

Table 5.1 Entropy of different window sizes performed in [102]

Where:

- W is the window size (based on number of packets),
- H_N is the entropy in normal condition,
- H_A is entropy during an attack,
- S_N and S_A are the standard deviation of entropy for normal and attack traffic conditions respectively.
- z is the test of significance. It represents the validity of the hypothesis between two averages of different populations. When the value is greater than 1.64, the hypothesis can be considered as

valid. In Table 1, it can be seen that for a window size of 50, z = 1.70. The value is computed using Equation 4:

$$z = \frac{\overline{H}_N - \overline{H}_A}{\sqrt{\frac{\sigma_n^2}{n}} + \sqrt{\frac{\sigma_r^2}{r}}} \tag{4}$$

Where:

- σ_n and σ_r are the same as S_N and S_A
- n is the population of normal traffic packets (n is not given)
- r is set to 25.

To test the hypothesis, a one-sided test of significance with 5% confidence interval was used. The formula for that test is shown in Equation 5:

$$\frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}} \tag{5}$$

Where:

- x is the mean of the population
- μ_0 is the sample mean,
- σ is the standard deviation and
- n is the sample count

Instead of doing this in our solution, we run several attacks to choose a threshold within the difference between H_N and H_A.

According to the Table 5.1, the entropy of attack is higher than that of normal condition. In this paper the entropy is based on the destination port and source IP address and because of that it is higher (the attack packets have different IP source addresses and they are, most likely, spoofed). In normal conditions, when attack is not occurred a connection between source and destination is established. The packets in these types of flows have the same IP source address. By reason of these packets have a greater number, the entropy of normal condition is lower and if attack occurs these several new source IP addresses increases entropy.

Qin et al [99] propose an algorithm with a window size 0.1 seconds and three levels of threshold. This method is concerned with avoiding false positive and false negatives in the network. However, as the authors themselves claim, the method is very time consuming and uses a lot of resources.

Ra et al. [100] propose a faster way of computing Entropy by basing the calculation on packet type and the volume of packets. This algorithm uses a time period window as well. For the threshold, the authors compute different datasets to find an applicable threshold and it is a multiple of standard deviation of entropy values. With this approach they reached higher the false negatives than other methods and lower false positives. There are no mention of resources used for fast computation and also no percentage of accuracy.

Despite of the lack of research of DDoS detection methods in SDN and VANET networks some methods are already proposed for these type of communication. Next we are looking into them.

In [103] is presented a machine learning method to learn the performance of the network and based on that, decide whether an attack occurs or not. This method is, largely, used in non-SDN networks and when used in this type of network it follows the same procedure and does not take into account the effect of DDoS on the controller. The system has to run alongside SDN and has to be trained for couple of hours before it can be used in the network. One other drawback is the fact that SDN may reconfigure the network frequently. This means the Self-Organizing Maps (SOM) solution has to be trained again for better protection. Finally, as the network expands, the neurons of the SOM algorithm have to increase resulting in expansive neurons in the network and a lightweight solution turns into a heavy drag for the network.

Shin et al. [104] propose Openflow for finding the shortest path to Network Intrusion Detection System (NIDS) devices. The solution requires the addition of NIDS devices along the links of the network to monitor traffic for suspicious activity. That can be seen as an issue. Entropy does not have these limitations. In the solution that we are going to propose we are using the controller itself for the detection of any attack.

In [105] DDoS detection method, which improves detection accuracy by using adaptive threshold algorithm is presented. To improve computational complexity, they use fast entropy approach on flow based data rather than conventional entropy approach. Fast Entropy and flow-based analysis show significant reduction in the performance in order to reduce the computational time while keeping good detection accuracy.

The system detects a DDoS attack when the difference between entropy of flow count at each instant and mean value of entropy in that time interval is greater than the threshold that is updated adaptively based on current traffic pattern.

This method is based on three objectives:

1. Flow aggregation for doing flow based attack detection.
2. Fast Entropy computation to detect DDoS attack with less computation time.
3. Adaptive Threshold Algorithm to improve detection accuracy.

In [106] an Attack Detection in VANET networks is proposed. They estimate the vulnerabilities of the communication protocols, which facilitate the implementation of the attacks. More precisely they consider the vulnerabilities of MAC and physical layers protocols. This approach has been tested in the case of greedy behaviour and it is able to detect the violation of the proper use of the CSMA/CA protocol rules. The detection idea is based on the supervision of the entropy of a chosen type of transmitted packets in the network. To calculate the entropy they supervise and determine the packets emission probabilities for the chosen type of packets (e.g Data and Ack). Given that the probability of emission of packets changes, then the entropy change, and the distinction between a normal network and a network under attack can be performed by the calculation of the packets entropy in each case.

5.6 ENTROPY BENEFITS FOR SDN BASED NETWORKS

Before discussing the solution we will explain briefly why entropy is convenient way to measure statistics in SDN based network. We have mentioned before, the main reason for choosing Entropy - its ability to measure randomness in a network and also we believe that the instant network status, typical to SDN networks, can help the controller to decide if the attack has been occurred or not.

In OF networks the controller is the operating system and losing it means losing the advantages of broad control with SDN. The reason why packets come to the controller is that the source IP address are new. There has not been an instance of them in the table of the switch, therefore they are passed to the controller. We already explained in the first section how for every new incoming connection, the controller is installing a flow in the switch, so that the rest of the incoming packets will be directed to the destination without further processing.

Also we know for each new packet in the controller the destination host is in the network of the controller. The network consists of the switches (they are RSUs and Base stations in our case) and hosts (for our topology they are vehicles) that are connected to it. With that knowledge (the packet in the controller is always new and the destination is in the network) the level of randomness can be quantified by calculating the entropy based on a window size. We already have explained that the window size is the number of incoming new packets that are used for calculating entropy. In the general case, maximum entropy occurs when each packet is destined to exactly one host. Minimum entropy will occurs when all the packets in a window are destined for a single host.

Being able to determine the randomness and have min and max based on entropy makes it a suitable method for DDoS detection in SDN. In that way, it is possible to see that the value of the entropy drops when a large number of packets are attacking one host or a subnet of hosts.

5.7 PROPOSED TOPOLOGY AND SOLUTION

Our solution hardly relies on the proposed method in [79]. We are going to perform a DDoS detection scheme using Entropy and we will take into account all parameters defined in that paper. Furthermore, we are going to implement the essence of that solution for SDN-based VANET-network. Therefore, we have to consider carefully the parameters of the network according to VANET standards and its limitations. We will modify this existing solution in order to provide

DDoS detection method SDN-based VANET. The main purpose of our work is to investigate DDoS attack case using UDP packets to provide high QoS which satisfied the needs of real-time services, such as accident prevention, traffic jam warning, or communication. We propose a V2I streaming on Mininet platform to build a V2I network and evaluate the performance of DDoS detection algorithm for real-time multimedia streaming under UDP on VANET.

As it will be further elaborated in the following sections, for the purpose of our work we will take into account very small packet windows size and therefore we assume that the topology can be treated as static. We are going to examine instant characteristics, that are not dependable on the mobility and also we are interested in really small time intervals. The communication between the vehicles (V2V) is data plane communication and it is not part of our interest in this work, we will guide our effort into another direction – the communication among the vehicles, base stations and the controller. As we already know, the RSUs and vehicles can act like switches and hosts at the same time. For that reason we will test our scenario with two different topologies.

In the first and third test cases we will perform network with one base station, 6 RSUs and 36 vehicles. The base station and the RSUs will act like switches and the vehicles will be hosts. In the second and third test cases we chose to create a topology with 40 vehicles, 4 RSUs and we will include 1 base station connected to the controller. That results in a topology, with 44 hosts, 1 switch and 1 controller.

In the next Section 6, subsection RESULTS AND TOPOLOGIES we described in details our test scenarios. The following figure 5.1 represent the basic topology that our solution relies on.

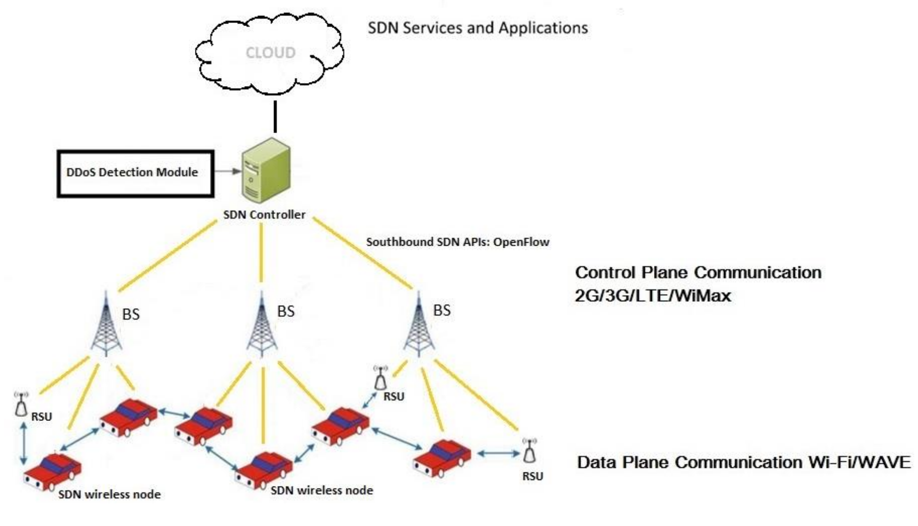

CLOUD
SDN Services and Applications
DDoS Detection Module
SDN Controller
Southbound SDN APIs: OpenFlow
Control Plane Communication
2G/3G/LTE/WiMax
BS
BS
BS
RSU
RSU
RSU
Data Plane Communication Wi-Fi/WAVE
SDN wireless node
SDN wireless node

Figure 5.1 Basic topology of proposed solution

5.7.1 CHOOSING WINDOW SIZE AND LOWEST ATTACK TRAFFIC THAT CAN BE DETECTED

It is very easy to change window size in the controller and this flexibility is the advantage of SDN. In [79] has chosen the window size to be 50. One reason is that in SDN, once a connection is established, the packets will not pass through the controller unless there is a new request. The other reason is the limited number of switches and hosts which can be connected to the controller. The third reason is the fact that a list of 50 values can be computed much faster than 500 and, an attack in a 50-packet window is detected earlier and easier.

In [79] the author also tested the entropy with four different window sizes and measured the CPU and memory usage load on the utilized SDN controller. Table 5.2 represents that there is no difference in memory usage but CPU usage increases with window size.

W	CPU1/CPU2	Memory usage
20	60% / 67%	1.2 GB
50	62% / 67%	1.2 GB
100	64% / 68%	1.2 GB
500	65% / 68%	1.2 GB

Table 5.2 Window size comparison

Table 5.3 shows the difference in entropy and the number of attack packets from each window size. The lowest attack traffic rate that the solution can detect with accuracy - 25%. Last column in the table represents the number of malicious packets when the attack traffic is 25% of all incoming packets.

W	H_N	H_A	H_N - H_A	H_A Packets 25%
20	1.22	1.1	0.12	5
50	1.5	1.3	0.2	12
100	1.6	1.4	0.2	25
500	1.66	1.47	0.19	125

Table 5.3 Comparison of five windows

According to the results in the tables it can be said in a window of size equal to 20, the difference of entropies is less than 10%, which makes choosing a threshold very difficult. With only five packets, probabilities of false positives will increases. It is noticeable the window size 500 does not bring a better difference of entropies and takes a much longer time than a window of 50 to compute entropy. H_A – H_B is 0.19 for window size with 500 packets which is 11% drop in normal traffic entropy. H_A – H_B in the window size of 50 is 0.12 which 10% drop in normal traffic entropy. This minor difference of 1% does not justify choosing a 10 times bigger window size.

Also, this table shows that a window sized of 50 and 100 look similar. The number of hosts in our topology is less than 100, we will chose 50.

According to all these experiments we decided to use window size equal to 30, and the main reason for that is in our scenario the vehicles will be 36. Proportionally window size with 30 packets and 36 hosts will result in the same accuracy of computation the entropy as network topology with 64 hosts and 50 packets window. We will focus our work into a really small time interval, because in our case the topology is treated as static, but in real world scenario VANETs topologies are dynamic, because of the movements of the cars. We believe that according to our experiments, a dynamic model of computation of the entropy can be developed. SDN controller can check periodically the number of hosts that are connected to the network and based on that information it can adjust the threshold. After that this threshold will be used to compare the entropy in the current time interval.

5.7.2 ENTROPY COMPUTATION IN SDN CONTROLLER

One of the main functions of the controller is collecting statistics from all switches to detect inactive flows – this is called time-out and these flows are removed if they don't receive any packets for a period of time (these parameter is mentioned in section 1 of our work). This time-out parameter can be set easily to different values. We are adding another set of statistics to the controller. In SDN, the packets are trying to establish a connection to the destination by getting a flow rule in the controller. Thus, having different source IP addresses is a known fact that does not help in the detection of DDoS. In SDN is known the exact number of hosts and switches connected to the controller. Also if we know the window size, the maximum entropy of the destination IP address is also clear – we can observe it when each packet is destined to exactly one host. After that we have to calculate the number of packets that are targeting a specific host or a subnet. In our proposed solution against DDoS attack, destination IP address is used for entropy computation.

The function that we are going to add to our controller will determine if a higher than normal rate of incoming packets destined to the same destination. We already choose the window size to be 30. The assumption is that the network has 30 or more hosts connected to it (in our case the vehicles are 36). In that new function, every 30 Packet_In messages will be parsed for their destination IP address and the entropy of that list of packet will be computed.

The next step is comparing the entropy to a predefined appropriate threshold, which will be discussed in the next chapter. We will determine this threshold based on our experiments. If the entropy is less than this threshold and it lasts for a minimum of 5 entropy periods in a row, it will be considered an attack. Detection within 5 entropy periods is 150 packets in the attack, which gives the network and early alert of attack. In the next chapter we will show the tests for choosing this periods.

We are going to mark the IP addresses of all hosts connected to the network with I (Equation 6):

$$I = \{x_1, x_2, x_3 .. x_N\} \tag{6}$$

W is the window which contains new packets' destination IP address x and their number of occurrence y (Equation 7 and Equation 8):

$$W = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots\} \tag{7}$$

$$x \subseteq I \tag{8}$$

The entropy is at its maximum when each destination IP address is unique (Equation 9). If the equation does not hold, then some of the IP addresses are appearing more than once.

$$\forall x \in \mathrm{W} : H_{MAX} \Longrightarrow y = 1 \tag{9}$$

In our solution we choose two conditions to be the trigger for an attack - threshold and the continuity of the attack. In [79] the authors propose a limit for the number of consecutive low entropy windows to avoid false positives. They can be caused by some glitches in the network that leads to irregularity in normal traffic. If a link to a switch goes down or some hosts become temporarily unavailable, the entropy might fall and trigger a false positive. The condition for declaring an attack is shown in Equation 10:

$$\begin{cases} attack\ H_A < T \wedge S \\ \neg attack, otherwise \end{cases} \tag{10}$$

Where:

- T is the threshold;
- S is an array of 5 windows with lower than T entropy;
- ¬ is the sign of negation.

An attack happened if entropy of attack H_A, is smaller than the threshold and, having five consecutive lower than threshold entropies is true.

Figure 5.2 shows the flowchart of the detection method:

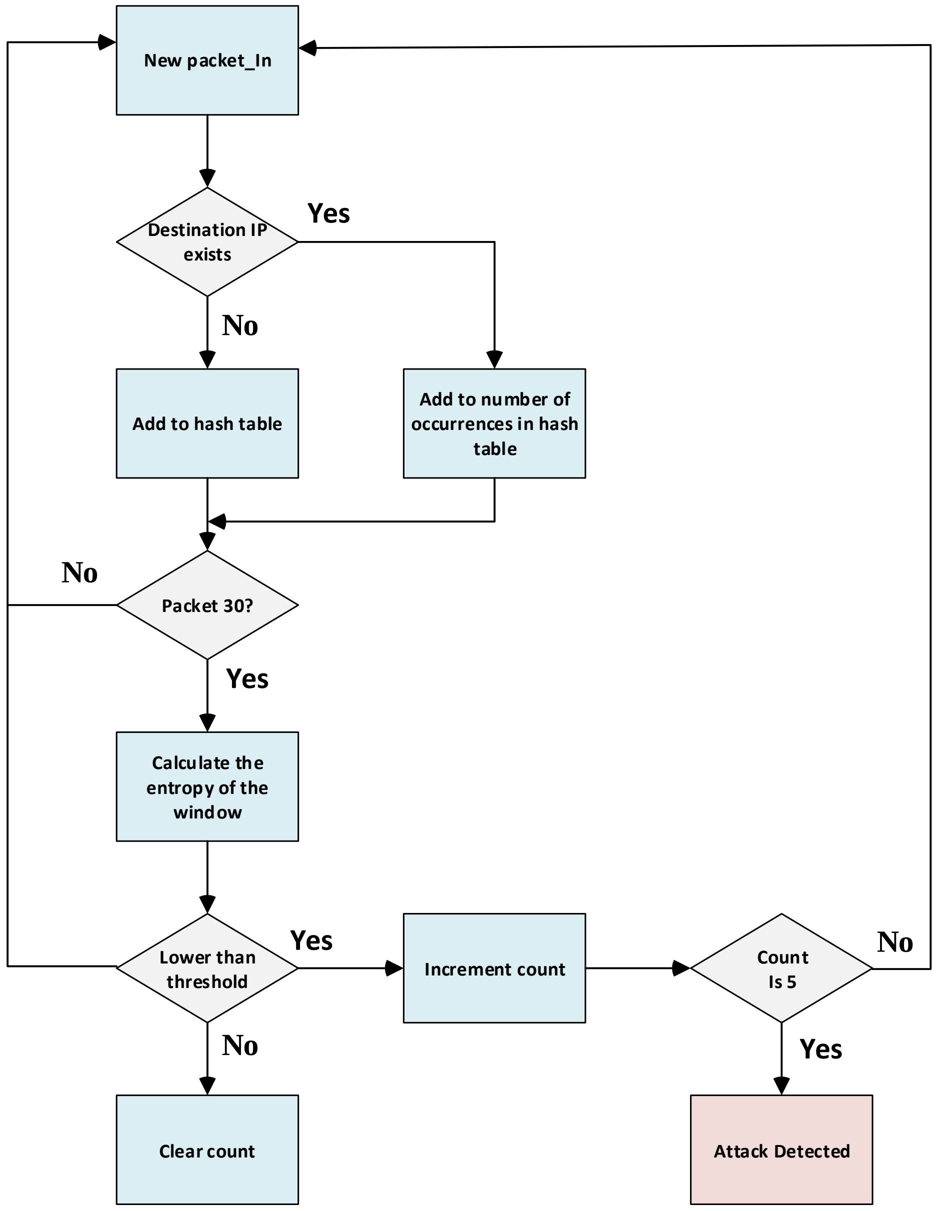


Figure 5.2 Flowchart of the proposed detection method

The initial step is "Packet_In" which represents that a new packet has arrived with new source address. Firstly, the destination IP address is examined to see if the destination IP address has an instance in the window. If yes, the system is increasing the count for that IP address, if not, it will be added as a new IP address. After that, the program cheeks if the window is full – the 30th packet has been arrived. If the window is full, the entropy will be computed. The last important step is comparing the entropy to the threshold. If it is higher than that threshold, the program returns to step one and wait for new packets (the count for consecutive lower-than-threshold entropies is set to 0). If the entropy is not higher than the threshold, the count for the consecutive lower-than threshold entropies is increased and if the count is equal to five, the traffic is classified as an attack.

This algorithm is done using two additional functions is the controller. First function is called when a new Packet_In message arrives and it accepts the destination IP address as an argument. The purpose of the second function is to computes the entropy. We has been included these functions in Appendix B.

6 SIMULATIONS AND RESULTS

6.1 CONTROLLER

The first part of our experiment is determining a type of controller. There are several popular controllers available. The one that is used in our solution is POX [107]. POX is extensively used for experiments, it is fast, lightweight and designed as a platform, so a custom controller can be built on top of it. POX provides a framework for communicating with SDN switches using either the OpenFlow or OVSDB protocol. It officially supports Windows, MAC OS and Linux but it has been used on other systems as well. It is a development version of its ancestor NOX, and both are running on Python. POX can be immediately used as a basic SDN controller by using the stock components that come bundled with it. More complex SDN controller can be develop by creating new POX components.

For entirety, three other controllers must be cited. Floodlight [108] is another broadly used controller, which is open-source and written in Java. Some advantages of the Floodlight are that it is designed to allow third parties to simply modify the software and develop applications and Representational state transfer (REST) APIs are involved to simplify the application interfaces to the product. Beacon [107] is another Java-based controller, open-source, which has high throughput and low latency. OpenDaylight [109] controller is the most latter addition to Openflow controllers. OpenDaylight announced its first release: Hydrogen. It was the first simultaneous release of OpenDaylight and features three different editions to help users get started: the Base Edition, the Virtualization Edition, and the Service Provider Edition.

6.2 NETWORK EMULATOR

Mininet [110] is the network emulator that we used in our work for running the experiments. It is the standard network emulation tool that can be used for SDN. Network namespace gives personal processes with their own network interfaces, routing tables and Address ARP tables. Mininet takes advantage of this feature of the kernel. It uses process-based virtualization to run hosts and switches on a single operating system (OS) kernel. Large networks (up to 4096 host on a single operating system) with different topologies can be tested and emulated. It is commonly achievable to develop a Mininet network simulates a hardware network, or a hardware network that simulates a Mininet network, and to compute the same application and binary code on either platforms.

Building a network in Mininet is as simple of entering the command *mn* to have network, which includes one switch connected to two hosts and one reference controller. The command is presented in Appendix A. NOX is the default controller of Mininet.

6.3 PACKET GENERATOR

Packet generation is performed by Scapy [111]. It is a very powerful packet manipulation tool for packet generation, sniffing, scanning, trace routing, probing, attacking and packet forging, written in Python. Scapy is used in our work to generate UDP packets and spoof the source IP address of the packets.

The POX controller use a Python programming language. Python is also used for generating random source IP address and host IP address. The function “random” inherits by “randrange” function is used. This function return a next random floating point number in the scope [0.0, 1.0). The core generator Mersenne Twister, used by Python generate a float of 53-bit precision and has a period of $2^{19937-1}$ [112]. To generate random numbers with even distribution we can use this number, which shows a long period of random generation. To model spoofed source IP address these numbers are united together. The interval of packet generation and type of the packets are two other parameters that we determined in Scapy. For both attack and normal traffic we used UDP packets. To fulfil the test case scenario the interval was tuned. For example, for an attack with 30% rate, normal traffic interval is 0.1 seconds and attack traffic is 0.03. This leads us to a window with 30% of packets intended to a single host. In Appendix D and E we presented the code for generating the normal and attack traffic correspondingly. The attack traffic generated to subnet of 6 hosts is presented in Appendix F.

6.4 NETWORK SETUP

The experiment was done on a Dell laptop with an Intel(R) Core(TM) i7-5500U CPU 2.40GHz processor, 2.4 GHz of power, 16GB of RAM, and 10/100/100/1000Mbitps network interface. The operating system is Linux Ubuntu 14.04 and Mininet version 2.2.1 was run native on Windows 8.1. Mininet 2.2.1 supports Openflow version 1.3.

6.5 NETWORK TOPOLOGIES

For the first test case we have been performed our tree-type network of depth two with 7 switches and 36 hosts by using Mininet. Figure 6.1 depict the network. For network switches we used Open Virtual Switch (OVS) [113]. It is a software switch that runs both on hardware and software. There is no difference between OVS and Openflow for our work. There are both supported by Mininet and both do the equivalent job. The L3_learning module of POX was used for the controller. Two function has been added to this module to use them for statistic collection and entropy computation. Later in this chapter, the experiments are presented in order to obtain information and based on it we will determine the threshold.

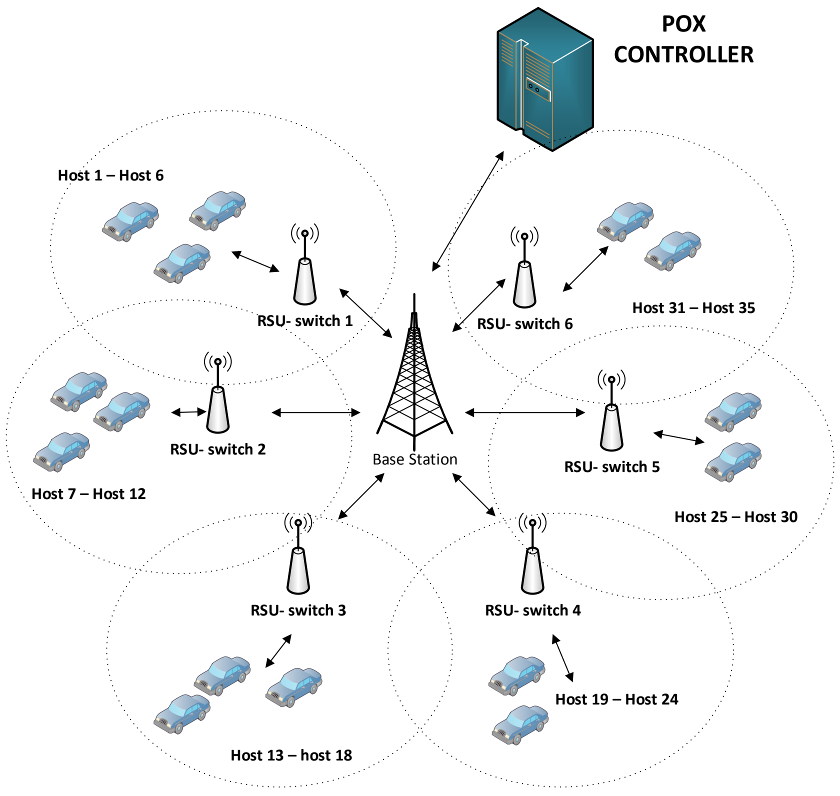


Figure 6.1 Topology for the first experiment

We decided to examine the values of the entropy of destination IP addresses in another scenario, because we want to see how the entropy will change in different topology. This is a real VANET case, in VANET scenario only RSUs and Base stations are static. The host move from one place to another and subsequently it can be connected to different nodes in the network. We believe that if the network can obtain a current status about the nodes and switches connected to it, the controller can perform a dynamic algorithm to detect DDoS attack. It can adjust the parameters of that method in real-time scenario. In our opinion, this is possible due to SDN controller and its benefits. We already discussed them in the previous sections. Our second topology differ from the first one by the number of hosts and switches that we had used.

In this second test case we will examine how the network performs when RSUs are part of the data plane communication, i.e. they will act like hosts. In this scenario the vehicles and the RSUs will be hosts and the base stations will act like switches. We chose the number of vehicles to be 40, number of RSUs to be 4 and we included 1 base stations connected to the controller. That results in a topology, with 44 hosts, 1 switch and 1 controller. In section two we examined several control modes in SDN-based VANET scenario. For this test case we chose central control mode. This is the operation mode, where all the operations of underlying SDN wireless nodes and RSUs are controlled by SDN controller. All the actions that the SDN data element implements are specifically determined by the SDN controller. The following figure 6.2 represents our second topology:

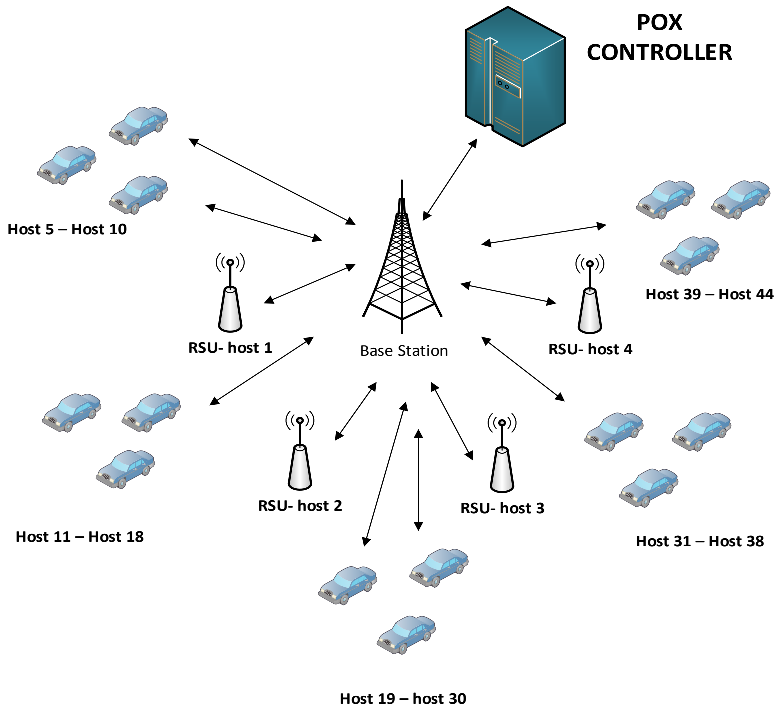


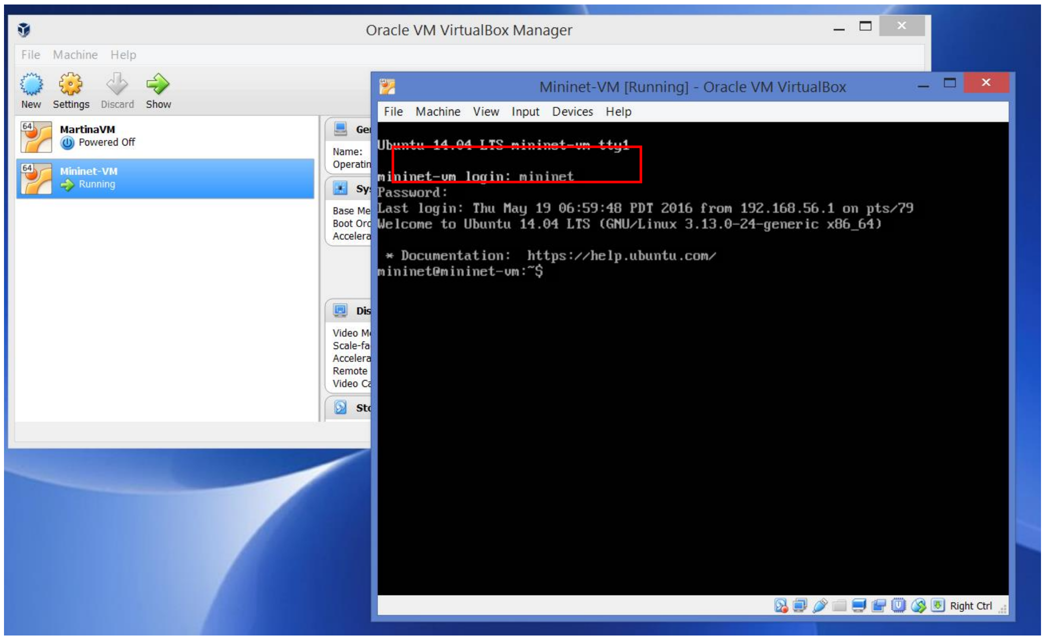
Figure 6.2 Topology for the second experiment

6.6 EXPERIMENTS AND RESULTS

The experiments includes two big test cases, but each of them consist of three sub-cases of attacks and a normal traffic runs. We also will include one experiment on how the entropy of destination IP address will change if the attack is launched by a Botnet of 3 hosts. We will launch DDoS attack in order to see how the entropy will be influenced by the attack. During all the experiments we are running two Scapy programs. The one that is generating the attack sends the packets faster than the one which generates normal traffic. Three different intensity attacks has been run on one host and one intensity attack on 6 hosts connected to the same switch and subnet. Normal traffic has been run on all switches with randomly generated packets going to all hosts. Codes that generated normal and attack traffic were started manually. The IP address in Mininet for all hosts are assigned gradually from 10.0.0.1. For test cases with attack traffic we will choose randomly one host to launch the attack to one and to a subnet of 6 hosts. The destination port is 80, the type of the attack is DDoS and all the traffic packet will be UDP. The packet header is only send to the controller by default in Openflow, so no payload was added to the generated packets. In the next subsections the parameters of the experiments are described in details and the results of them are presented and analysed.

6.6.1 WORKING WITH THE EMULATOR, POX CONTROLLER AND PACKET GENERATOR

On a Virtual Machine (VM) we installed Mininet emulator for prototyping of SDN-based VANET network.

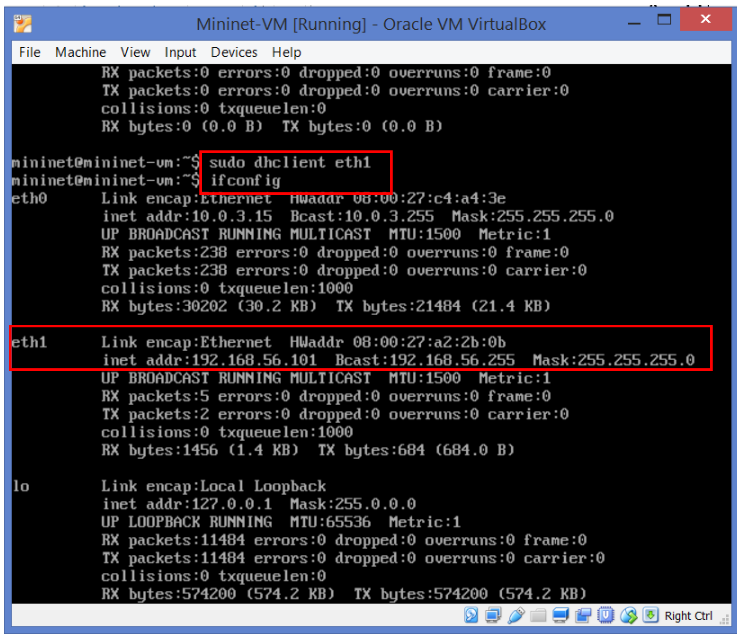

Oracle VM VirtualBox Manager
File Machine Help
New Settings Discard Show
MartinaVM
Powered Off
Mininet-VM
Running
Mininet-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Ubuntu 14.04 LTS mininet-vm tty1

mininet-vm login: mininet
Password:
Last login: Thu May 19 06:59:48 PDT 2016 from 192.168.56.1 on pts/79
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
mininet@mininet-vm:~$
```

On the next picture we are using "dhcliend" command that provides a means for configuring one or more network interfaces using the Dynamic Host Configuration Protocol, BOOTP protocol, or if these protocols fail, by statically assigning an address.


Mininet-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

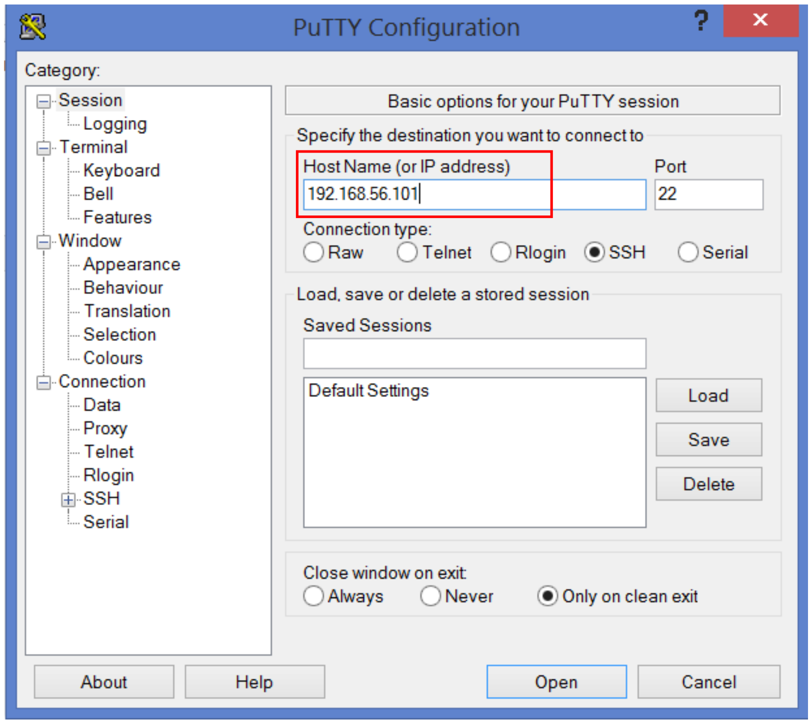
```
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet@mininet-vm:~$ sudo dhclient eth1
mininet@mininet-vm:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:c4:a4:3e
          inet addr:10.0.3.15  Bcast:10.0.3.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:238 errors:0 dropped:0 overruns:0 frame:0
          TX packets:238 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:30202 (30.2 KB)  TX bytes:21484 (21.4 KB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:a2:2b:0b
          inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1456 (1.4 KB)  TX bytes:684 (684.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:11484 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11484 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:574200 (574.2 KB)  TX bytes:574200 (574.2 KB)
```

We used the IP address of interface eth1 to open our virtual machine using Putty, because for the purpose of our work, we needed to use the graphical interface Xming and this is one way to do that.


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.56.101

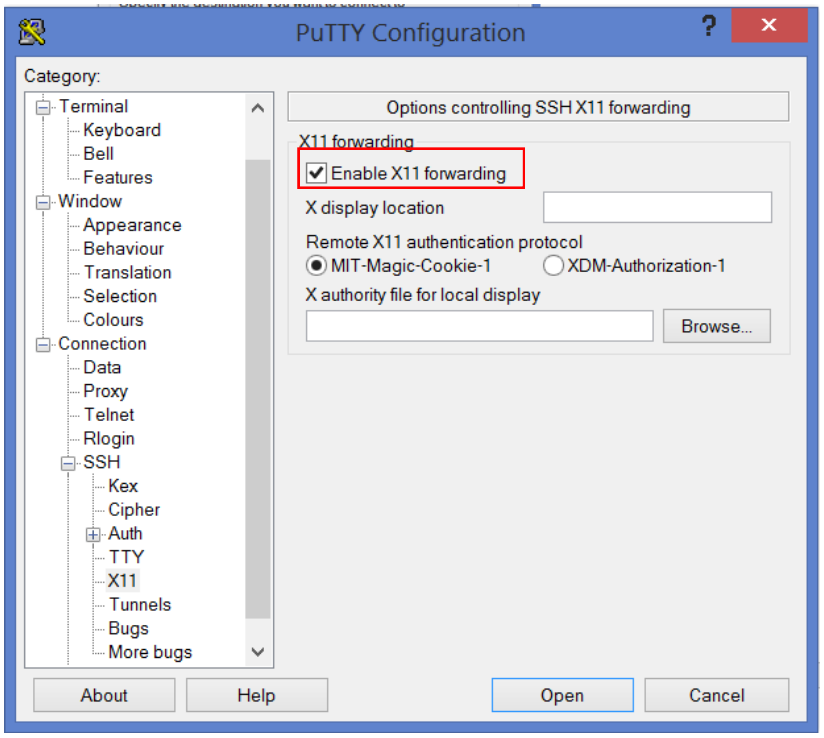
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: Always Never [✓ Only on clean exit]


PuTTY Configuration

Options controlling SSH X11 forwarding

X11 forwarding

[✓ Enable X11 forwarding]

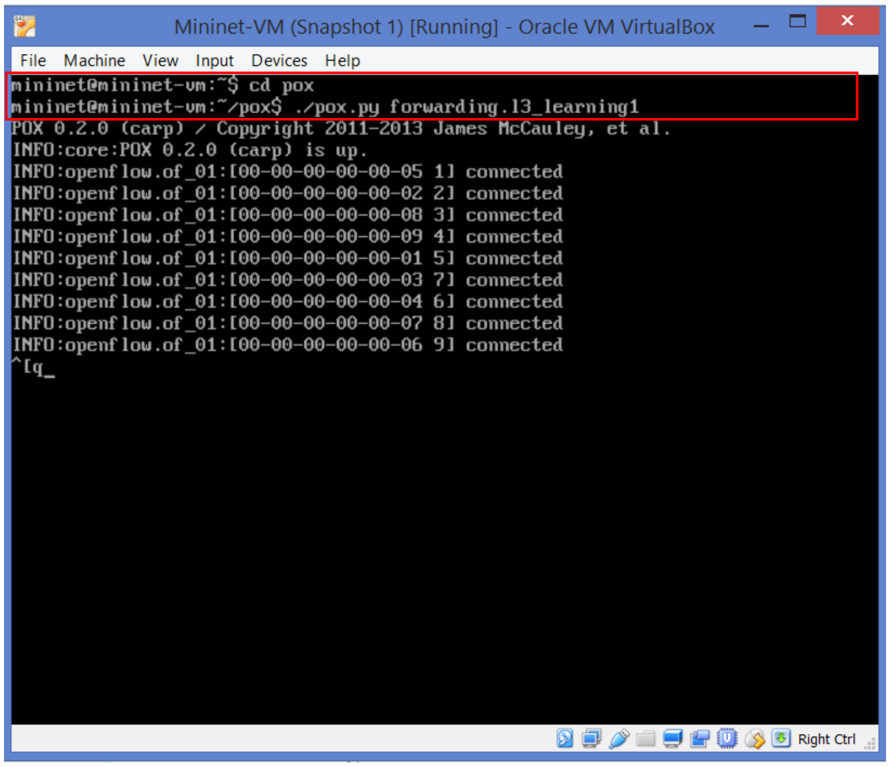
X display location:

Remote X11 authentication protocol: [✓ MIT-Magic-Cookie-1] XDM-Authorization-1

X authority file for local display:

The POX controller was installed on the same virtual machine. On the next picture we started the POX controller with the modified component - l3_learning. We used that module to run POX's learning switch. POX comes with a number of stock components, they provide core functionality or convenient features. Two additional functions for statistic collection and entropy computation are added here. We have called that modified component l3_learning1.

./pox.py forwarding.l3_learning1


Mininet-VM (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
mininet@mininet-vm:~$ cd pox
mininet@mininet-vm:~/pox$ ./pox.py forwarding.l3_learning1
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[00-00-00-00-00-05 1] connected
INFO:openflow.of_01:[00-00-00-00-00-02 2] connected
INFO:openflow.of_01:[00-00-00-00-00-08 3] connected
INFO:openflow.of_01:[00-00-00-00-00-09 4] connected
INFO:openflow.of_01:[00-00-00-00-00-01 5] connected
INFO:openflow.of_01:[00-00-00-00-00-03 7] connected
INFO:openflow.of_01:[00-00-00-00-00-04 6] connected
INFO:openflow.of_01:[00-00-00-00-00-07 8] connected
INFO:openflow.of_01:[00-00-00-00-00-06 9] connected
^[q_
```

After running the POX controller, in another window Mininet topology has been created. For the purpose of our work we used the following command to create the first topology:

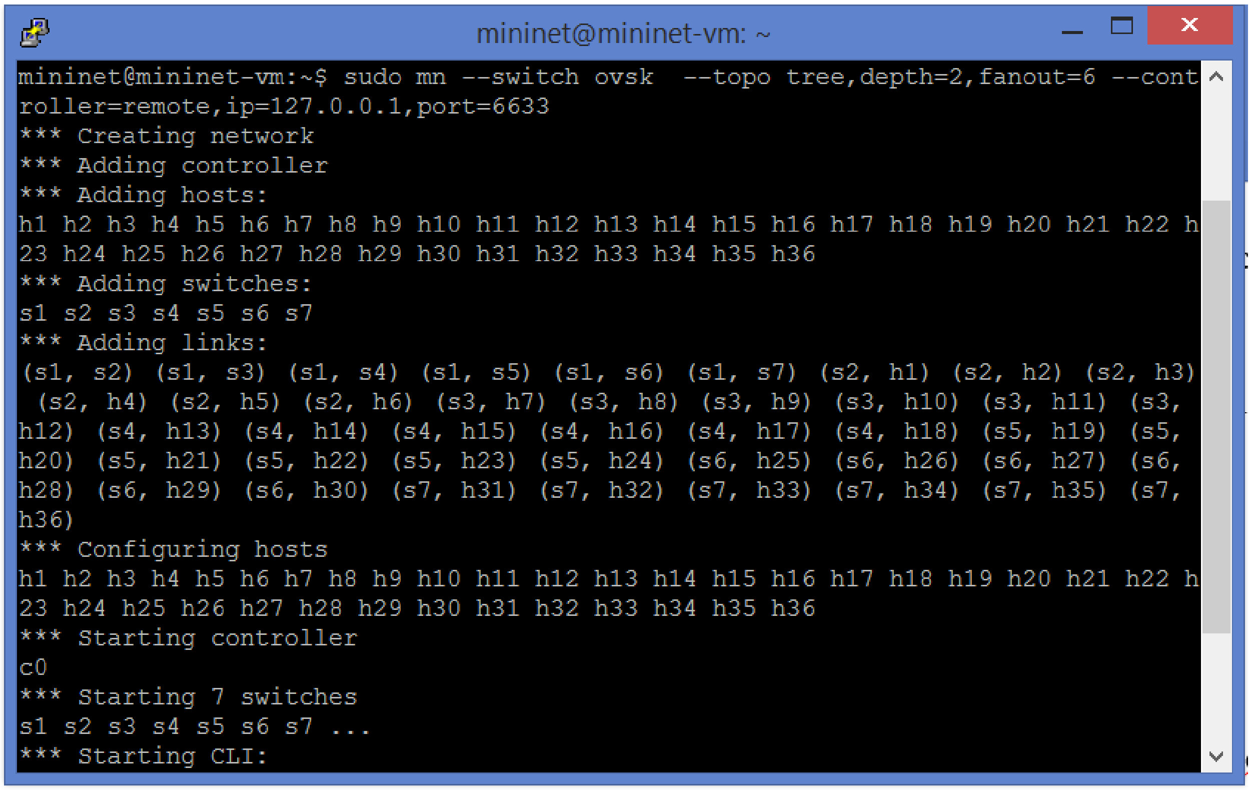
sudo mn --switch ovsk --topo tree, depth = 2, fanout = 6 –controller = remote,ip = 127.0.0.1, port= 6633

-- *topo tree, depth =2* – This results in creating a tree topology on two layers – the base station (s1) is on the first layer and the RSUs (s2 – s7) on the second. Overall 7 switches.

-- *fanout = 6* – That results in 36 vehicles connected to the switches.

-- *switch ovsk* - We used Open vSwitch (OVS), which comes preinstalled on the Mininet VM. In our scenario every RSU will act like a switch.

-- *controller = remote* - Start up Mininet to connect to the "remote" POX controller. It actually running locally, but outside of Mininet's control.

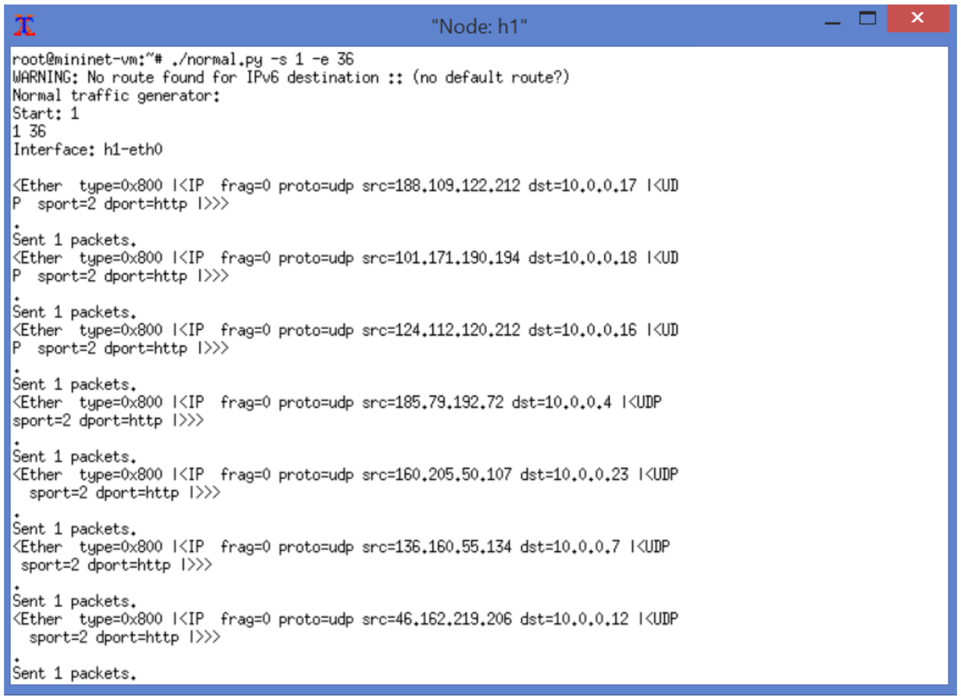


```
mininet@mininet-vm:~$ sudo mn --switch ovsk  --topo tree,depth=2,fanout=6 --cont
roller=remote,ip=127.0.0.1,port=6633
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h
23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s1, s5) (s1, s6) (s1, s7) (s2, h1) (s2, h2) (s2, h3)
 (s2, h4) (s2, h5) (s2, h6) (s3, h7) (s3, h8) (s3, h9) (s3, h10) (s3, h11) (s3,
h12) (s4, h13) (s4, h14) (s4, h15) (s4, h16) (s4, h17) (s4, h18) (s5, h19) (s5,
h20) (s5, h21) (s5, h22) (s5, h23) (s5, h24) (s6, h25) (s6, h26) (s6, h27) (s6,
h28) (s6, h29) (s6, h30) (s7, h31) (s7, h32) (s7, h33) (s7, h34) (s7, h35) (s7,
h36)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h
23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
```

6.6.2 GENERATION OF TRAFFIC IN MININET

The attributes that are defined when the function is started are the last numbers of the destination IP addresses. The code will generate traffic choosing a destination IP address between these numbers. On the picture also can be observed the spoofed source IP address. These packets are sent using interface eth0. Here we launched normal traffic between all the hosts in the network. We randomly chose one host and open it via *xterm* command and after that we run the script to generate the normal traffic

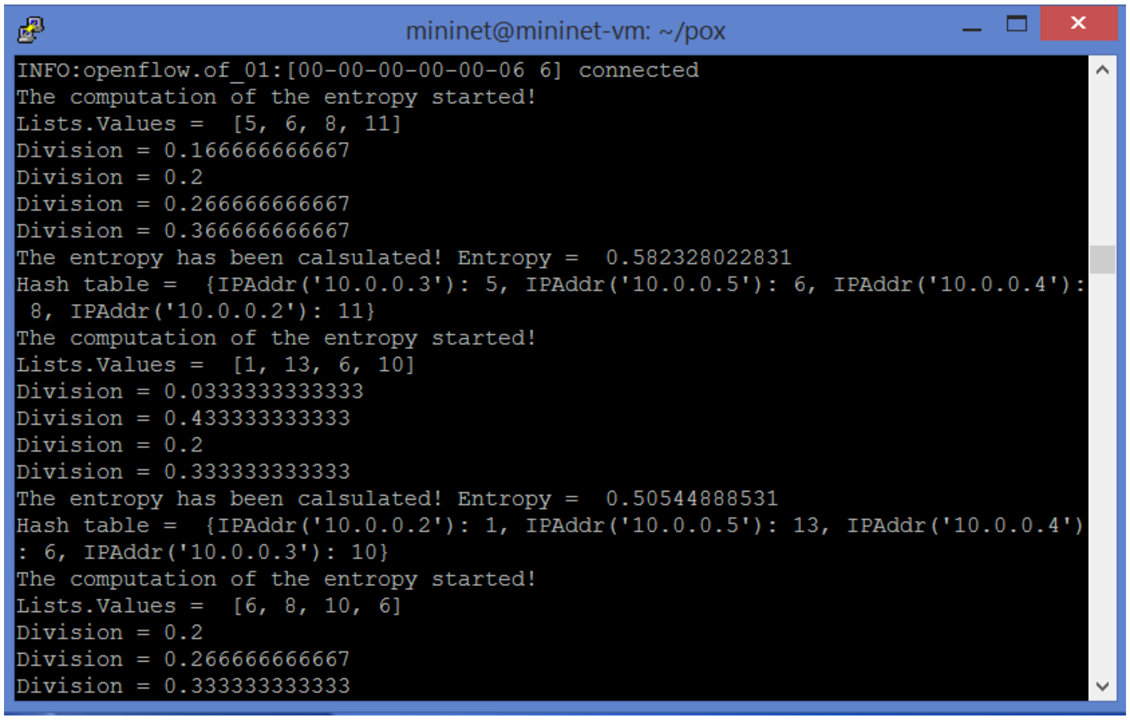
./normal.py –s 1 –e 36



```
root@mininet-vm:~# ./normal.py -s 1 -e 36
WARNING: No route found for IPv6 destination :: (no default route?)
Normal traffic generator:
Start: 1
1 36
Interface: h1-eth0

<Ether  type=0x800 |<IP  frag=0 proto=udp src=188.109.122.212 dst=10.0.0.17 |<UD
P  sport=2 dport=http |>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=udp src=101.171.190.194 dst=10.0.0.18 |<UD
P  sport=2 dport=http |>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=udp src=124.112.120.212 dst=10.0.0.16 |<UD
P  sport=2 dport=http |>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=udp src=185.79.192.72 dst=10.0.0.4 |<UDP
sport=2 dport=http |>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=udp src=160.205.50.107 dst=10.0.0.23 |<UDP
  sport=2 dport=http |>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=udp src=136.160.55.134 dst=10.0.0.7 |<UDP
 sport=2 dport=http |>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=udp src=46.162.219.206 dst=10.0.0.12 |<UDP
  sport=2 dport=http |>>>
.
Sent 1 packets.
```

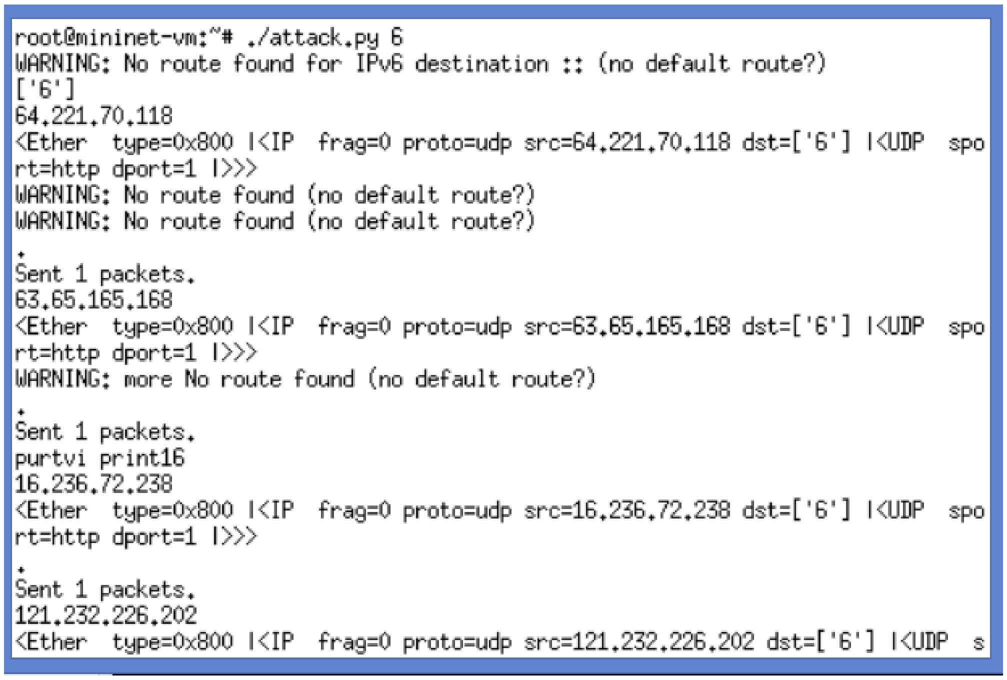
The next picture shows how the traffic is treated by the controller. When the traffic generation started the functions of the controller begins to collect information about the packets and based on that it computes the entropy.



```
INFO:openflow.of_01:[00-00-00-00-00-06 6] connected
The computation of the entropy started!
Lists.Values =  [5, 6, 8, 11]
Division = 0.166666666667
Division = 0.2
Division = 0.266666666667
Division = 0.366666666667
The entropy has been calsulated! Entropy =  0.582328022831
Hash table =  {IPAddr('10.0.0.3'): 5, IPAddr('10.0.0.5'): 6, IPAddr('10.0.0.4'):
 8, IPAddr('10.0.0.2'): 11}
The computation of the entropy started!
Lists.Values =  [1, 13, 6, 10]
Division = 0.0333333333333
Division = 0.433333333333
Division = 0.2
Division = 0.333333333333
The entropy has been calsulated! Entropy =  0.50544888531
Hash table =  {IPAddr('10.0.0.2'): 1, IPAddr('10.0.0.5'): 13, IPAddr('10.0.0.4')
: 6, IPAddr('10.0.0.3'): 10}
The computation of the entropy started!
Lists.Values =  [6, 8, 10, 6]
Division = 0.2
Division = 0.266666666667
Division = 0.333333333333
```

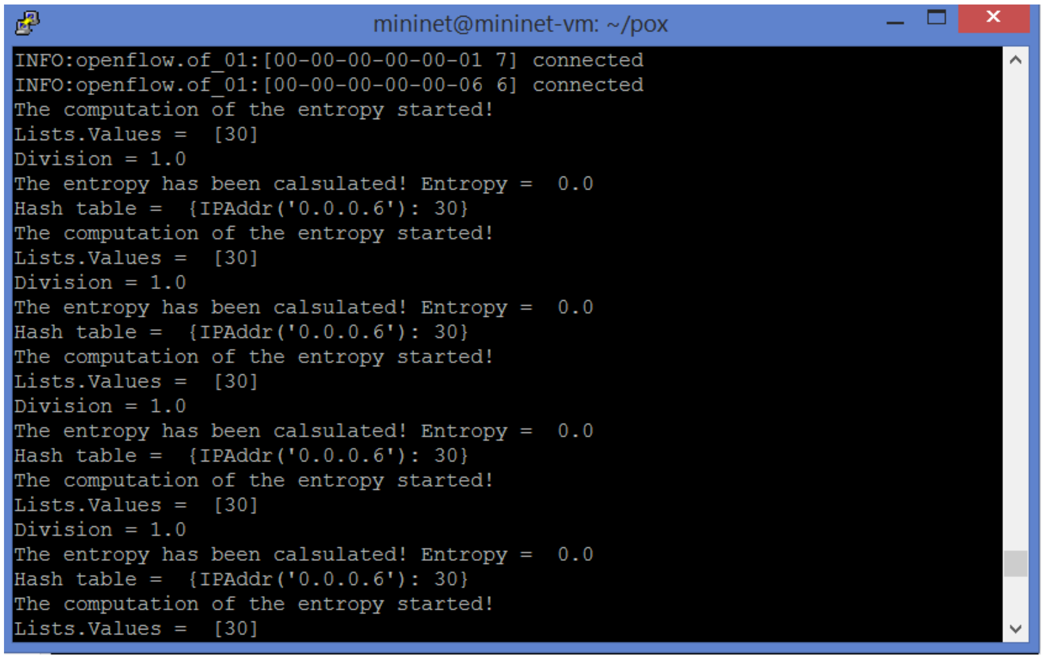
On the next picture an example how attack traffic to host number 4 can be generated from another host.

./attack.py 4



```
root@mininet-vm:~# ./attack.py 6
WARNING: No route found for IPv6 destination :: (no default route?)
['6']
64.221.70.118
<Ether  type=0x800 |<IP  frag=0 proto=udp src=64.221.70.118 dst=['6'] |<UDP  spo
rt=http dport=1 |>>>
WARNING: No route found (no default route?)
WARNING: No route found (no default route?)
.
Sent 1 packets.
63.65.165.168
<Ether  type=0x800 |<IP  frag=0 proto=udp src=63.65.165.168 dst=['6'] |<UDP  spo
rt=http dport=1 |>>>
WARNING: more No route found (no default route?)
.
Sent 1 packets.
purtvi print16
16.236.72.238
<Ether  type=0x800 |<IP  frag=0 proto=udp src=16.236.72.238 dst=['6'] |<UDP  spo
rt=http dport=1 |>>>
.
Sent 1 packets.
121.232.226.202
<Ether  type=0x800 |<IP  frag=0 proto=udp src=121.232.226.202 dst=['6'] |<UDP  s
```

We included the next picture here just to see how launching attack traffic only on one host will result in Entropy = 0



```
INFO:openflow.of_01:[00-00-00-00-00-01 7] connected
INFO:openflow.of_01:[00-00-00-00-00-06 6] connected
The computation of the entropy started!
Lists.Values =  [30]
Division = 1.0
The entropy has been calsulated! Entropy =  0.0
Hash table =  {IPAddr('0.0.0.6'): 30}
The computation of the entropy started!
Lists.Values =  [30]
Division = 1.0
The entropy has been calsulated! Entropy =  0.0
Hash table =  {IPAddr('0.0.0.6'): 30}
The computation of the entropy started!
Lists.Values =  [30]
Division = 1.0
The entropy has been calsulated! Entropy =  0.0
Hash table =  {IPAddr('0.0.0.6'): 30}
The computation of the entropy started!
Lists.Values =  [30]
Division = 1.0
The entropy has been calsulated! Entropy =  0.0
Hash table =  {IPAddr('0.0.0.6'): 30}
The computation of the entropy started!
Lists.Values =  [30]
```

6.6.3 TEST CASE 1 – THE TOPOLOGY CONSISTS 36 VEHICLES, 6 RSUs AND 1 BASE STATION

6.6.3.1 CALCULATING AVERAGE ENTROPY VALUE OF NORMAL TRAFFIC

To calculate the average entropy value in normal circumstances, we launched normal traffic in the whole network, between all hosts and switches. We have been performed 5 runs, each of them consists of 750 packets in window size equal to 30. The traffic interval is 0.1 seconds. This experiment covers observation of 3750 packets. The results are presented in next figures and the obtained information is summarized and analyzed in Table 6.1 and Figure 6.3.

The following graph represents the fluctuations of the entropy in 25 consistent windows of 30 packets. According to the graph it can be concluded that the entropy varies between 0.68 and 1.07.

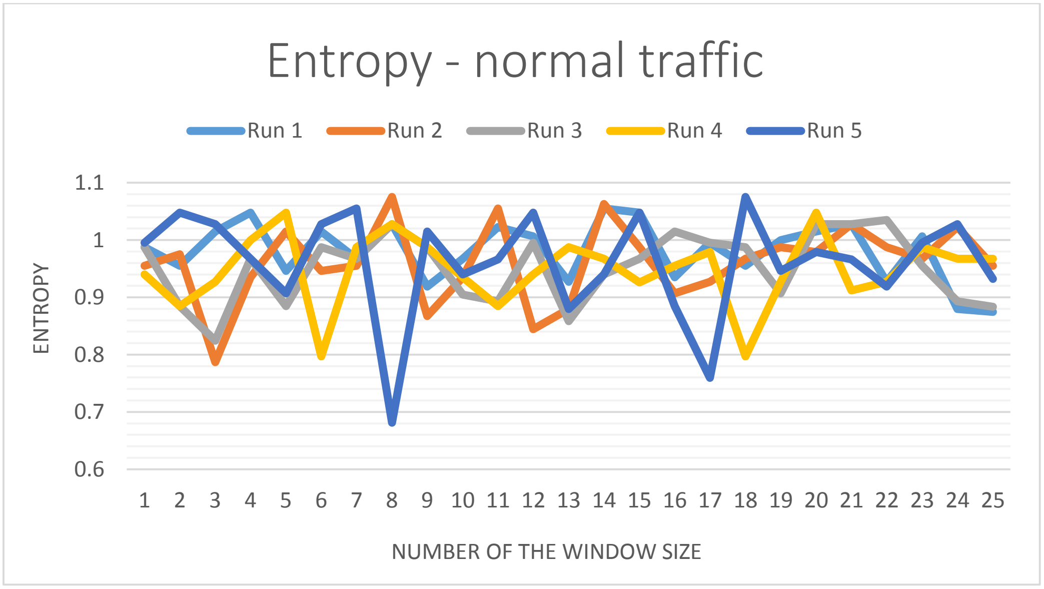


Figure 6.3 Entropy - normal traffic

The next Table 6.1 summarizes the information obtained by this experiment. The entropy in each window for every run can be observed. We calculated the average value for each run and at the end the average entropy for the whole experiment is presented.

Average Entropy value for normal traffic					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	0.987506126	0.954943586	0.987506126	0.939794001	0.995080919
window no. 2	0.954943586	0.975012253	0.884507083	0.884507083	1.047712125
window no. 3	1.015149585	0.786819463	0.824301084	0.927300127	1.027643459
window no. 4	1.047712125	0.93487492	0.96743746	1	0.96743746
window no. 5	0.94636671	1.015149585	0.884507083	1.047712125	0.907231461
window no. 6	1.015149585	0.94636671	0.987506126	0.796657624	1.027643459
window no. 7	0.96743746	0.954943586	0.96743746	0.987506126	1.055286918
window no. 8	1.027643459	1.075355585	1.027643459	1.027643459	0.681164707
window no. 9	0.918723251	0.867094128	0.987506126	0.987506126	1.015149585
window no. 10	0.96743746	0.93487492	0.904575749	0.93487492	0.939794001
window no. 11	1.022724378	1.055286918	0.893315627	0.884507083	0.966435377
window no. 12	1.006572709	0.84436975	0.995080919	0.939794001	1.047712125
window no. 13	0.927300127	0.879588002	0.858517252	0.987506126	0.879588002
window no. 14	1.055286918	1.062861711	0.939794001	0.96743746	0.939794001
window no. 15	1.047712125	0.987506126	0.96743746	0.926298044	1.047712125
window no. 16	0.93487492	0.907231461	1.015149585	0.954943586	0.884507083
window no. 17	0.995080919	0.927300127	0.995080919	0.97892925	0.759176003
window no. 18	0.954943586	0.96743746	0.987506126	0.796657624	1.075355585
window no. 19	1	0.987506126	0.907231461	0.927300127	0.94636671
window no. 20	1.015149585	0.97892925	1.027643459	1.047712125	0.97892925
window no. 21	1.027643459	1.027643459	1.027643459	0.912150542	0.966435377
window no. 22	0.927300127	0.987506126	1.035218252	0.927300127	0.918723251
window no. 23	1.006572709	0.96743746	0.954943586	0.987506126	0.995080919
window no. 24	0.879588002	1.022724378	0.893315627	0.96743746	1.027643459
window no. 25	0.874668921	0.954943586	0.883504999	0.96743746	0.932219208
Average value	0.980939513	0.960148267	0.95217242	0.948256749	0.961192903
For the whole experiment	**0.96054197**				

Table 6.1 Average entropy value for normal traffic

6.6.3.2 CALCULATING AVERAGE ENTROPY FOR NETWORK UNDER ATTACK

In this part of our experiment we launched normal traffic in the whole network, between all hosts and switches with traffic interval equal to 0.1. At the same time we launched attack traffic to one host. As we mentioned before the source IP address is spoofed and usually the DDoS attacks have shorter traffic interval. We has been performed 2 test cases for attack traffic directed to one host. In the first one we launched the attack with interval 0.03 seconds and in the second one we launched the attack with interval 0.05 seconds.

6.6.3.2.1 Traffic interval = 0.03 seconds

We observed 5 runs, each of them consists of 750 packets with window size equal to 30. This experiment covers observation of 3750 packets. The results are presented and information is summarized in Table 6.2 and Figure 6.4.

On the following graph we can observe how the entropy varies in 25 windows of 30 packets. The drop at the begging is caused by the fact we ran the attack traffic manually after the second window. Therefore we are not taking this values into account. With that assumption we can conclude that the lowest point that the entropy can reach is 0.53 and the highest is 0.78.

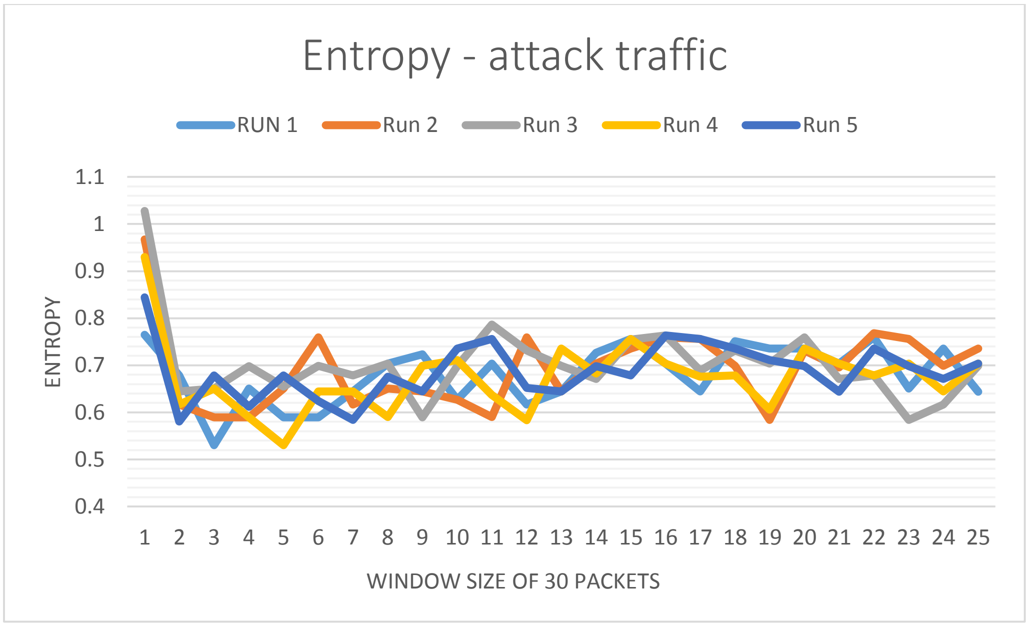


Figure 6.4 Entropy – attack traffic, interval 0.03 seconds

Table 6.2 represents the information obtained by this test case. Similarly to the first results here can be observed the entropy in each window for every run. The last rows contain information about the average value for each run and the average entropy for the whole test case.

Average Entropy value for network under attack, attack traffic interval 0.3 sec					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	0.764746629	0.96743746	1.027643459	0.929955839	0.84436975
window no. 2	0.678158457	0.616661823	0.644075429	0.616661823	0.580821375
window no. 3	0.530103	0.589018364	0.650514998	0.650514998	0.678158457
window no. 4	0.650514998	0.589018364	0.698227123	0.589018364	0.611512889
window no. 5	0.589018364	0.650514998	0.654506426	0.530103	0.678158457
window no. 6	0.589018364	0.759176003	0.698781796	0.644075429	0.624006762
window no. 7	0.644075429	0.616661823	0.678158457	0.644075429	0.58386943
window no. 8	0.703700877	0.650514998	0.703700877	0.590308999	0.675577176
window no. 9	0.723289301	0.644075429	0.589018364	0.698970004	0.644075429
window no. 10	0.626862967	0.626862967	0.698227123	0.71127567	0.735783175
window no. 11	0.703700877	0.590308999	0.786819463	0.63673049	0.755851841
window no. 12	0.616661823	0.759176003	0.731344336	0.583288879	0.651650221
window no. 13	0.644305282	0.644075429	0.698970004	0.735783175	0.644075429
window no. 14	0.726613463	0.703700877	0.671138337	0.682630128	0.698227123
window no. 15	0.755851841	0.735783175	0.754256923	0.755851841	0.678158457
window no. 16	0.703700877	0.759176003	0.763426634	0.703700877	0.763426634
window no. 17	0.644075429	0.755851841	0.688592519	0.675577176	0.755851841
window no. 18	0.751413003	0.698970004	0.731344336	0.678158457	0.735783175
window no. 19	0.735783175	0.58386943	0.703700877	0.605458584	0.71127567
window no. 20	0.735783175	0.731344336	0.759176003	0.735783175	0.698227123
window no. 21	0.703700877	0.695645842	0.671138337	0.703700877	0.643494878
window no. 22	0.759176003	0.767865101	0.678158457	0.678158457	0.735783175
window no. 23	0.650514998	0.755851841	0.58386943	0.703700877	0.698227123
window no. 24	0.735783175	0.698970004	0.616661823	0.644075429	0.671138337
window no. 25	0.643494878	0.735783175	0.698970004	0.703700877	0.703700877
Average value	0.680401891	0.693052572	0.703216862	0.673250354	0.688048192
For the whole experiment	**0.687593974**				

Table 6.2 Average entropy value for network under attack, attack traffic interval 0.03 seconds

6.6.3.2.2 Traffic interval = 0.05 seconds

In the following figure 6.5 and Table 6.3 we obtained the average entropy for the second part of this experiment, we launched attack traffic with interval of 0.05 seconds. The other parameters remain the same, as these in the previous test case.

Figure 6.3 represents the measured entropy value in 25 consistent windows of 30 packets, when the network was under attack traffic with interval of the packets 0.05 seconds. Similarly to test case 1, we launched the attack manually after the 30th packet of the normal traffic, therefore we will not pay attention on these values. For that test case, when the attack traffic interval is half more than the normal traffic interval, we can observe lowest point at 0.61 and highest value of 0.84.

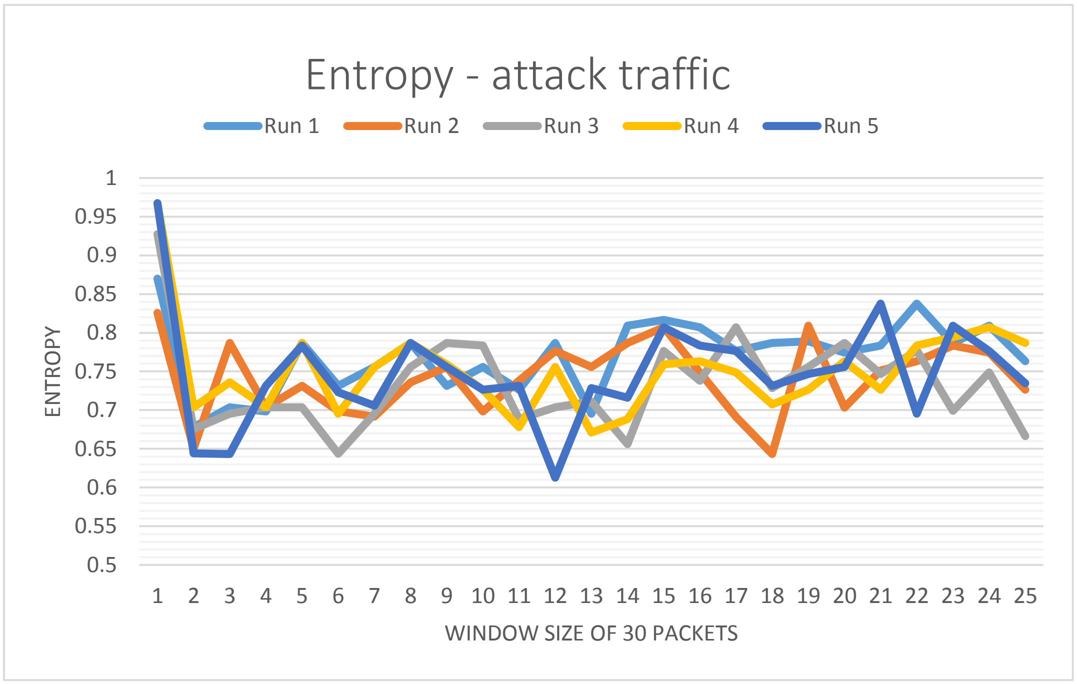


Figure 6.5 Entropy – attack traffic, interval 0.05 seconds

Table 6.3 represents the results obtained by these test case. The entropy in each window for every run are included. The average value for each run and the average entropy for the whole experiment are placed in the last rows.

Average Entropy value for network under attack, attack traffic interval 0.3 sec					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	0.86974984	0.82553484	0.927300127	0.96743746	0.9674375
window no. 2	0.678158457	0.650515	0.675577176	0.70370088	0.6440754
window no. 3	0.703700877	0.78681946	0.695645842	0.73578317	0.6434949
window no. 4	0.698227123	0.70370088	0.703700877	0.70370088	0.7313443
window no. 5	0.786819463	0.73134434	0.703700877	0.78681946	0.7834953
window no. 6	0.731344336	0.6987818	0.643683086	0.69564584	0.7232893
window no. 7	0.755851841	0.69178755	0.695645842	0.75585184	0.7058019
window no. 8	0.786819463	0.73578317	0.755851841	0.78681946	0.7868195
window no. 9	0.731344336	0.75585184	0.786819463	0.759176	0.7558518
window no. 10	0.755851841	0.69822712	0.7834953	0.72661346	0.7266135
window no. 11	0.726613463	0.73810525	0.688592519	0.67815846	0.7313443
window no. 12	0.786819463	0.77644198	0.703700877	0.75585184	0.6127155
window no. 13	0.695645842	0.75585184	0.71127567	0.67113834	0.7285263
window no. 14	0.809004517	0.78681946	0.656029979	0.68859252	0.716236
window no. 15	0.81657931	0.80688813	0.776441977	0.759176	0.8068881
window no. 16	0.806888129	0.74879852	0.738105254	0.76342663	0.7834953
window no. 17	0.776441977	0.691207	0.806888129	0.74879852	0.776442
window no. 18	0.786819463	0.64349488	0.728526294	0.70723923	0.7313443
window no. 19	0.788732293	0.80900452	0.755851841	0.72642526	0.7466821
window no. 20	0.774325589	0.70322063	0.786819463	0.76342663	0.7558518
window no. 21	0.7834953	0.751413	0.74668213	0.72661346	0.8374465
window no. 22	0.837446502	0.76342663	0.777240503	0.7834953	0.6956458
window no. 23	0.786819463	0.7834953	0.698970004	0.79439426	0.8090045
window no. 24	0.809004517	0.77432559	0.748798518	0.80688813	0.776442
window no. 25	0.763426634	0.72661346	0.666407464	0.78681946	0.7347811
Average value	0.769837202	0.741498088	0.734470042	0.7512797	0.74844276
For the whole experiment	**0.749105559**				

Table 6.3 Average entropy value for network under attack, attack traffic interval 0.05 seconds

6.6.3.3 CALCULATING AVERAGE ENTROPY VALUE FOR NETWORK UNDER ATTACK ON 6 HOSTS

In that part of the experiment we are going to examine what is the average entropy value for network under attack on a subnet of 6 hosts. The topology and the other parameters remain the same as the previous test case. We will launch DDoS attack on 6 hosts connected to one switch with traffic interval 0.03 seconds. At the same time normal traffic is running between all the hosts.

Next figure 6.6 represents how the entropy varies during the test. The lowest point is reached in the first run and it takes value of 0.52 and the highest point is in the third run with value of 0.94.

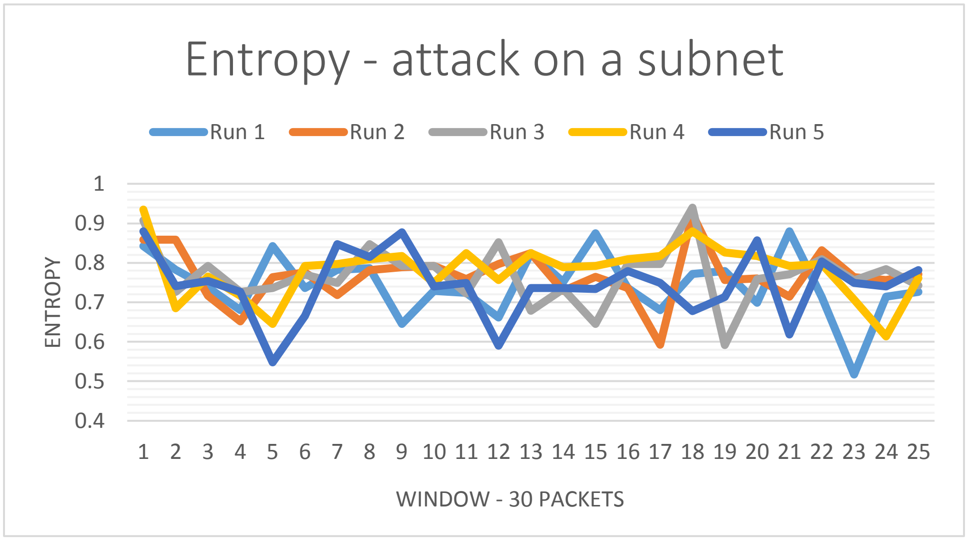


Figure 6.6 Entropy – attack traffic on a subnet, interval 0.03 seconds

The average value for each run in 25 windows of 30 packets is presented in Table 6.4. According to the results, we can conclude the average value for the entropy on a subnet of 6 hosts has closer value to the attack traffic on one host, but with highest interval of 0.05 seconds. As expected, the entropy on a subnet has higher value than the entropy on one host with the same attack traffic interval.

Average Entropy value for network under attack on 6 hosts					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	0.842106381	0.858517252	0.907231461	0.93487492	0.879588002
window no. 2	0.782159584	0.858517252	0.726613463	0.685054171	0.741370706
window no. 3	0.738105254	0.717034504	0.791738544	0.765328836	0.754256923
window no. 4	0.68013509	0.651909424	0.726613463	0.717034504	0.726613463
window no. 5	0.842106381	0.764095084	0.736451625	0.645595917	0.547908297
window no. 6	0.736451625	0.775586875	0.770247917	0.791738544	0.665868141
window no. 7	0.782741791	0.718036587	0.748798518	0.796657624	0.847025462
window no. 8	0.786819463	0.781900382	0.847025462	0.809543841	0.814462922
window no. 9	0.644916838	0.788732293	0.791738544	0.816957959	0.877163958
window no. 10	0.728876833	0.791738544	0.791738544	0.749337842	0.740341089
window no. 11	0.724350094	0.759176003	0.722535792	0.824301084	0.749337842
window no. 12	0.661488383	0.796657624	0.851944543	0.75675196	0.590308999
window no. 13	0.824301084	0.824301084	0.678158457	0.824301084	0.736451625
window no. 14	0.751601211	0.728526294	0.734027582	0.788732293	0.736451625
window no. 15	0.874668921	0.764746629	0.645595917	0.791738544	0.734027582
window no. 16	0.735783175	0.736451625	0.796657624	0.809004517	0.77924467
window no. 17	0.68013509	0.592646799	0.796657624	0.816957959	0.748798518
window no. 18	0.771522896	0.922381046	0.939794001	0.879588002	0.677899255
window no. 19	0.77924467	0.75675196	0.591703425	0.825954712	0.713727247
window no. 20	0.698970004	0.760409755	0.759176003	0.816957959	0.856863624
window no. 21	0.879588002	0.71411959	0.770667794	0.791738544	0.61880756
window no. 22	0.713727247	0.831875876	0.809543841	0.796657624	0.803230334
window no. 23	0.517144233	0.764095084	0.755749877	0.707239226	0.749337842
window no. 24	0.714378792	0.75675196	0.784395419	0.613888479	0.740341089
window no. 25	0.726613463	0.770667794	0.742836127	0.761671041	0.781900382
Average value	0.74471746	0.767425093	0.768705663	0.780704287	0.744453086
For the whole experiment	**0.761201118**				

Table 6.4 Average entropy value for network under attack on 6 hosts, attack traffic interval 0.03 seconds

6.6.3.4 CONCLUSION REMARKS

For each experiment we launched 3750 packets in 5 runs, each of them consisted of 750 packets and the window size was equal to 30. We collected information about the entropy in several different cases with different traffic intervals. We decided to summarize that information in the next table for further convenience.

	Average entropy value	**Lowest entropy value**	**Highest entropy value**
Normal traffic – 0.1 sec	0.96	0.68	1.07
Attack on one host - 0.03 sec	0.69	0.53	0.78
Attack on one host - 0.05 sec	0.75	0.61	0.84
Attack on 6 hosts - 0.03 sec	0.76	0.52	0.94

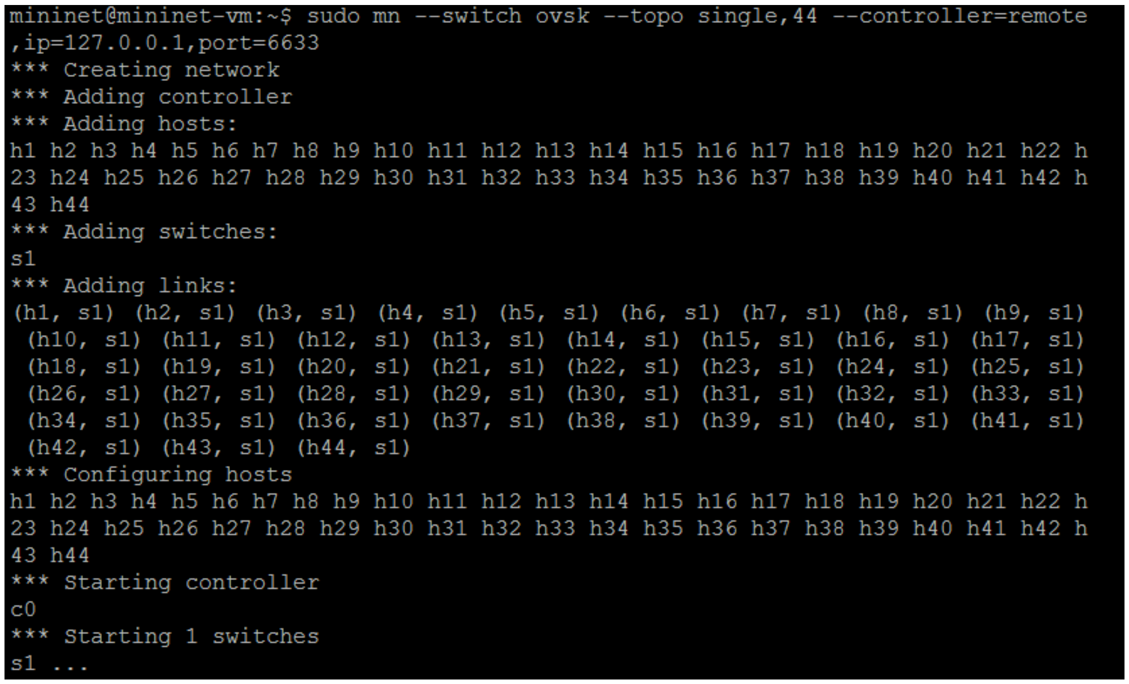
Table 6.5 Summarized information of test case 1

6.6.4 TEST CASE 2 – THE TOPOLOGY CONSISTS 40 VEHICLES, 4 RSUs AND 1 BASE STATION

6.6.4.1 CREATING THE TOPOLOGY IN MININET

In this part of the experiments we used the following command to create the topology in Mininet:

sudo mn --switch ovsk --topo single,44 --controller=remote,ip=127.0.0.1,port=6633



```
mininet@mininet-vm:~$ sudo mn --switch ovsk --topo single,44 --controller=remote
,ip=127.0.0.1,port=6633
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h
23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h
43 h44
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1)
 (h10, s1) (h11, s1) (h12, s1) (h13, s1) (h14, s1) (h15, s1) (h16, s1) (h17, s1)
 (h18, s1) (h19, s1) (h20, s1) (h21, s1) (h22, s1) (h23, s1) (h24, s1) (h25, s1)
 (h26, s1) (h27, s1) (h28, s1) (h29, s1) (h30, s1) (h31, s1) (h32, s1) (h33, s1)
 (h34, s1) (h35, s1) (h36, s1) (h37, s1) (h38, s1) (h39, s1) (h40, s1) (h41, s1)
 (h42, s1) (h43, s1) (h44, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h
23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h
43 h44
*** Starting controller
c0
*** Starting 1 switches
s1 ...
```

In the next subsections we will follow the same experiments as these in the first test case. Firstly, we will compute the entropy of destination IP addresses for the network only with normal traffic with interval of 0.1 seconds. After that, we will perform tests on the entropy when the attack occurs. We will use again different attack traffic intervals, 0.03 seconds and 0.05 seconds. The last experiment will give us information about how the entropy will change when a subnet of 6 hosts is under attack.

6.6.4.2 CALCULATING AVERAGE ENTROPY VALUE OF NORMAL TRAFFIC

We launched normal traffic in the whole network, between all hosts. We has been performed 5 runs, each of them consists of 750 packets in window size equal to 30. The traffic interval is 0.1 seconds. This experiment covers observation of 3750 packets. The results are presented in next figure and the obtained information is summarized and analyzed in Table 6.6 and Figure 6.7.

The obtained results about how the entropy fluctuates are presented in the next figure. According to the next graph it can be said that in network with only one switch and 44 hosts the lowest point that the entropy can reach is 1.17 and the highest is 1.39.

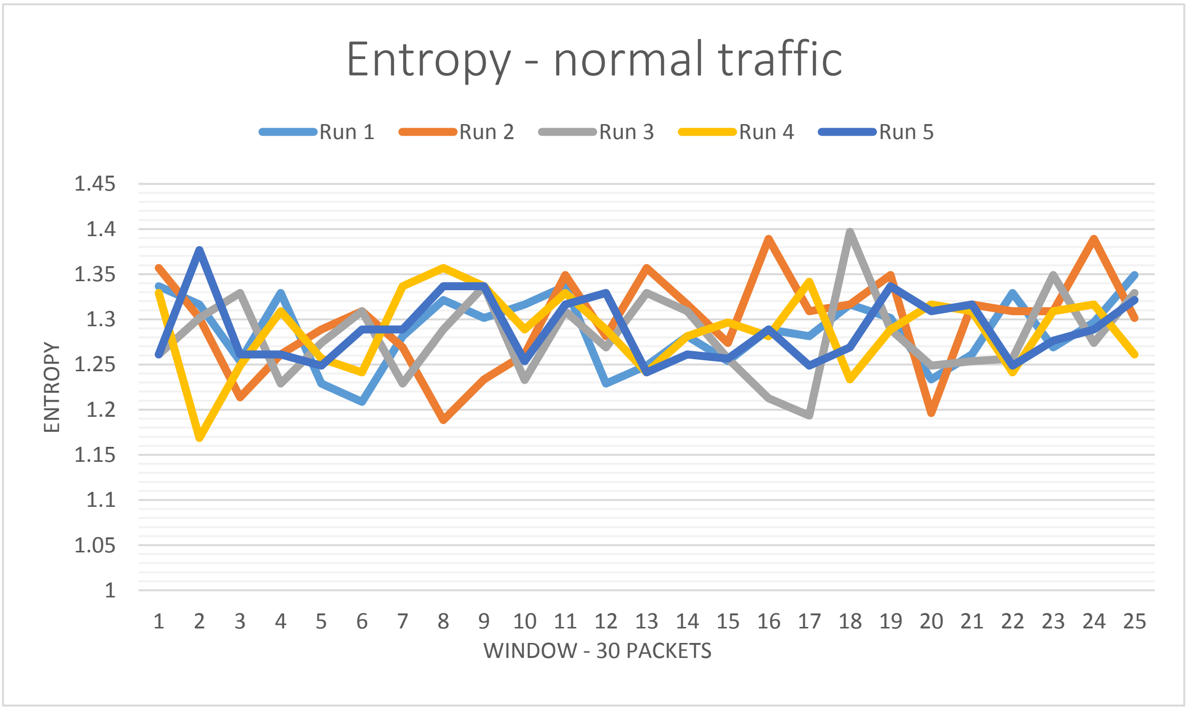


Figure 6.7 Entropy - normal traffic, test case 2

The next table gives us more detailed information about the entropy in normal circumstances. The entropy in each window, for every run can be observed. We calculated the average value for each run and at the end the average entropy for the whole experiment is presented.

Average Entropy value for network only with normal traffic					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	1.33664059	1.356709256	1.261285006	1.329065797	1.26128501
window no. 2	1.316571924	1.301422338	1.301422338	1.168516466	1.37677792
window no. 3	1.253710213	1.21357288	1.329065797	1.248791132	1.26128501
window no. 4	1.329065797	1.261285006	1.228722465	1.308997131	1.26128501
window no. 5	1.228722465	1.288928465	1.273778879	1.256365925	1.24879113
window no. 6	1.208653799	1.308997131	1.308997131	1.241216339	1.28892846
window no. 7	1.281353672	1.268859798	1.228722465	1.33664059	1.28892846
window no. 8	1.321491005	1.188585133	1.288928465	1.356709256	1.33664059
window no. 9	1.301422338	1.233641546	1.33664059	1.33664059	1.33664059
window no. 10	1.316571924	1.261285006	1.232639463	1.288928465	1.25371021
window no. 11	1.33664059	1.349134464	1.308997131	1.329065797	1.31657192
window no. 12	1.228722465	1.281353672	1.268859798	1.288928465	1.3290658
window no. 13	1.248791132	1.356709256	1.329065797	1.241216339	1.24121634
window no. 14	1.281353672	1.316571924	1.308997131	1.281353672	1.26128501
window no. 15	1.253710213	1.273778879	1.256365925	1.296503257	1.25636592
window no. 16	1.288928465	1.389271796	1.212570797	1.281353672	1.28892846
window no. 17	1.281353672	1.308997131	1.193504214	1.341559671	1.24879113
window no. 18	1.316571924	1.316571924	1.396846589	1.233641546	1.2688598
window no. 19	1.301422338	1.349134464	1.288928465	1.288928465	1.33664059
window no. 20	1.233641546	1.196159925	1.248791132	1.316571924	1.30899713
window no. 21	1.261285006	1.316571924	1.253710213	1.308997131	1.31657192
window no. 22	1.329065797	1.308997131	1.256365925	1.241216339	1.24879113
window no. 23	1.268859798	1.308997131	1.349134464	1.308997131	1.27643459
window no. 24	1.296503257	1.389271796	1.273778879	1.316571924	1.28892846
window no. 25	1.349134464	1.301422338	1.329065797	1.261285006	1.321491
Average value	1.286807523	1.297849213	1.282607394	1.288322481	1.288928465
For the whole experiment	**1.288903015**				

Table 6.6 Average entropy value for normal traffic - test case 2

6.6.4.3 CALCULATING AVERAGE ENTROPY FOR NETWORK UNDER ATTACK

In this part of our experiment we launched normal traffic in the whole network, between all hosts with traffic interval equal to 0.1 as in the first computation of the entropy. Meanwhile, we launched attack traffic to randomly chosen host. In this subsection are included two test cases for attack traffic directed to one host. In the first one we launched the attack with interval 0.03 seconds and in the second one we launched the attack with interval 0.05 seconds.

6.6.4.3.1 Traffic interval = 0.03 seconds

We observed 5 runs, each of them consists of 750 packets with window size equal to 30. This experiment also covers observation of 3750 packets. On the following graph we can observe how the entropy varies in 25 windows of 30 packets. The drop at the begging is caused by the fact we ran the attack traffic manually after the first window. Therefore, we are not taking this values into account. With that assumption we can conclude that the lowest point that the entropy can reach is 0.41 and the highest is 0.55.

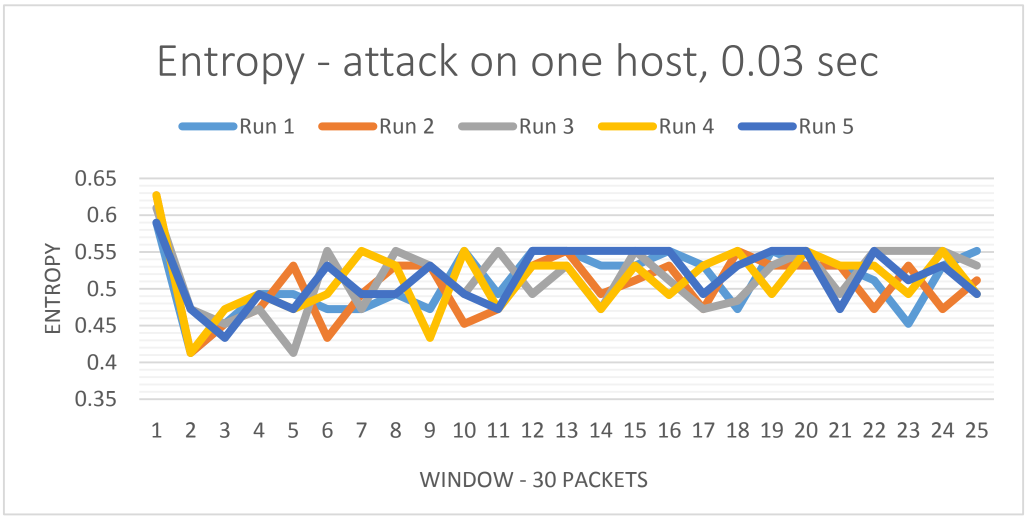


Figure 6.8 Entropy – attack traffic, interval 0.03 seconds, test case 2

Table 6.7 represents the information obtained by this test case. The last rows again contain information about the average value for each run and the average entropy for the whole test case. The entropy falls dramatically comparing to the first test case. One of the reasons is that we didn`t change the window size for this part of our experiment.

Average Entropy value for network under attack on 1 host, 0.03 sec					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	0.589699258	0.627106641	0.609767924	0.627106641	0.58969926
window no. 2	0.413061247	0.413061247	0.472609289	0.413061247	0.472609289
window no. 3	0.452540623	0.452540623	0.452540623	0.472609289	0.433129914
window no. 4	0.492677955	0.472609289	0.472609289	0.492677955	0.492677955
window no. 5	0.492677955	0.531499082	0.413061247	0.472609289	0.472609289
window no. 6	0.472609289	0.433129914	0.551567748	0.492677955	0.531499082
window no. 7	0.472609289	0.492677955	0.472609289	0.551567748	0.492677955
window no. 8	0.492677955	0.531499082	0.551567748	0.531499082	0.492677955
window no. 9	0.472609289	0.531499082	0.531499082	0.433129914	0.531499082
window no. 10	0.551567748	0.452540623	0.492677955	0.551567748	0.492677955
window no. 11	0.492677955	0.472609289	0.551567748	0.472609289	0.472609289
window no. 12	0.551567748	0.531499082	0.492677955	0.531499082	0.551567748
window no. 13	0.551567748	0.551567748	0.531499082	0.531499082	0.551567748
window no. 14	0.531499082	0.492677955	0.472609289	0.472609289	0.551567748
window no. 15	0.531499082	0.511430416	0.551567748	0.531499082	0.551567748
window no. 16	0.551567748	0.531499082	0.511430416	0.491361749	0.551567748
window no. 17	0.531499082	0.472609289	0.472609289	0.531499082	0.492677955
window no. 18	0.472609289	0.551567748	0.483786957	0.551567748	0.531499082
window no. 19	0.551567748	0.531499082	0.531499082	0.492677955	0.551567748
window no. 20	0.531499082	0.531499082	0.551567748	0.551567748	0.551567748
window no. 21	0.531499082	0.531499082	0.492677955	0.531499082	0.472609289
window no. 22	0.511430416	0.472609289	0.551567748	0.531499082	0.551567748
window no. 23	0.452540623	0.531499082	0.551567748	0.492677955	0.511430416
window no. 24	0.531499082	0.472609289	0.551567748	0.551567748	0.531499082
window no. 25	0.551567748	0.511430416	0.531499082	0.492677955	0.492677955
Average value	0.511152885	0.505450779	0.514008232	0.511872752	0.516772112
For the whole experiment	**0.511851352**				

Table 6.7 Average entropy value for network under attack, attack traffic interval 0.03 seconds, test case 2

6.6.4.3.2 Traffic interval = 0.05 seconds

In the following figure 6.9 and Table 6.8 we obtained the average entropy when we launched attack traffic with interval of 0.05 seconds to one randomly chosen host. The other parameters remain the same as in the previous test case with traffic interval 0.03 seconds.

Figure 6.9 represents the measured entropy value in 25 consistent windows of 30 packets, when the network was under attack traffic with interval of the packets 0.05 seconds. Similarly to previous experiment, we launched the attack manually after the 30th packet of the normal traffic. For the analysis we will ignore these values in the first window. For that test, we can observe lowest point at 0.53 and highest value 0.72.

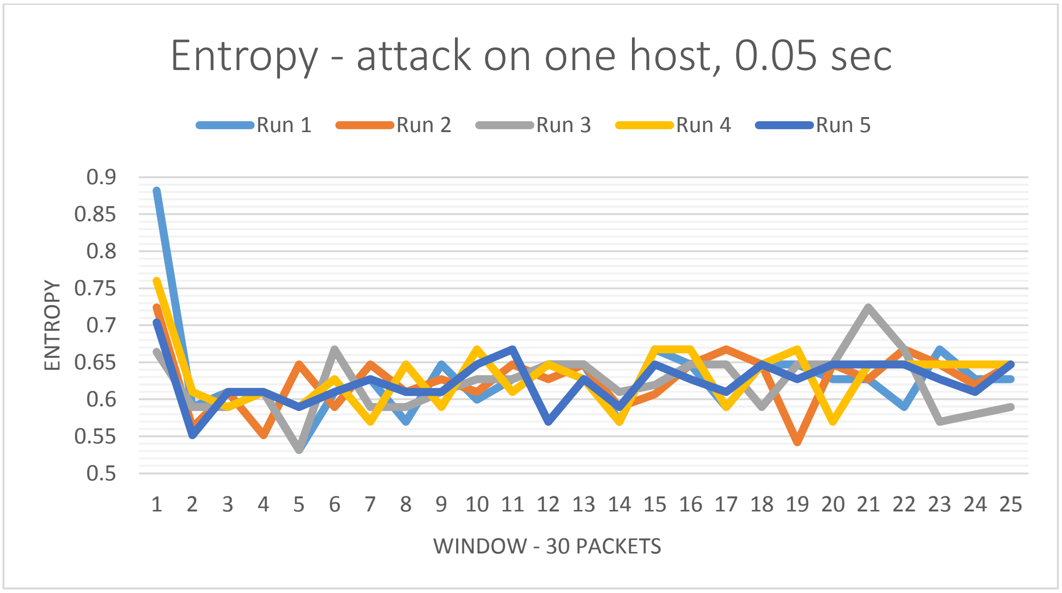


Figure 6.9 Entropy – attack traffic, interval 0.05 seconds, test case 2

Table 6.8 shows the detailed information about the entropy of destination IP addresses for this experiment. As expected, the average entropy for attack traffic with interval 0.05 is higher than the entropy obtained from the previous test case, when we used interval equal to 0.03 seconds.

Average Entropy value for network under attack on 1 host, 0.05sec					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	0.882055506	0.723957752	0.663751753	0.7597982	0.703889085
window no. 2	0.589699258	0.562055799	0.589699258	0.609767924	0.551567748
window no. 3	0.609767924	0.609767924	0.589699258	0.589699258	0.609767924
window no. 4	0.609767924	0.551567748	0.609767924	0.609767924	0.609767924
window no. 5	0.531499082	0.647175308	0.531499082	0.589699258	0.589699258
window no. 6	0.609767924	0.589699258	0.667243974	0.627106641	0.609767924
window no. 7	0.627106641	0.647175308	0.589699258	0.569630592	0.627106641
window no. 8	0.569630592	0.609767924	0.589699258	0.647175308	0.609767924
window no. 9	0.647175308	0.627106641	0.609767924	0.589699258	0.609767924
window no. 10	0.599463182	0.609767924	0.627106641	0.667243974	0.647175308
window no. 11	0.627106641	0.647175308	0.627106641	0.609767924	0.667243974
window no. 12	0.647175308	0.627106641	0.647175308	0.647175308	0.569630592
window no. 13	0.647175308	0.647175308	0.647175308	0.627106641	0.627106641
window no. 14	0.569630592	0.589699258	0.609767924	0.569630592	0.589699258
window no. 15	0.667243974	0.607037975	0.619531849	0.667243974	0.647175308
window no. 16	0.647175308	0.647175308	0.647175308	0.667243974	0.627106641
window no. 17	0.589699258	0.667243974	0.647175308	0.589699258	0.609767924
window no. 18	0.647175308	0.647175308	0.589699258	0.647175308	0.647175308
window no. 19	0.647175308	0.541987132	0.647175308	0.667243974	0.627106641
window no. 20	0.627106641	0.647175308	0.647175308	0.569630592	0.647175308
window no. 21	0.627106641	0.627106641	0.723957752	0.647175308	0.647175308
window no. 22	0.589699258	0.667243974	0.667243974	0.647175308	0.647175308
window no. 23	0.667243974	0.647175308	0.569630592	0.647175308	0.627106641
window no. 24	0.627106641	0.619531849	0.579394516	0.647175308	0.609767924
window no. 25	0.627106641	0.647175308	0.589699258	0.647175308	0.647175308
Average value	0.629234406	0.626209047	0.621080718	0.630295297	0.62423463
For the whole experiment	**0.626210819**				

Table 6.8 Average entropy value for network under attack, attack traffic interval 0.05 seconds, test case 2

6.6.4.4 CALCULATING AVERAGE ENTROPY VALUE FOR NETWORK UNDER ATTACK ON 6 HOSTS

This is the last part of our experiment on a network that consists 44 hosts and one switch. We will repeat the last test which was made for the previous topology. We are going to examine what is the average entropy value for network under attack on a subnet of 6 hosts. The other parameters remain the same as the previous test. We will launch DDoS attack on 6 hosts with traffic interval 0.03 seconds. At the same time normal traffic is running between all the hosts.

Next figure 6.10 depicts how the entropy varies during the test. The lowest point is reached in the first run and it takes value of 0.8 and the highest point is obtained in the third run with value of 1.03.

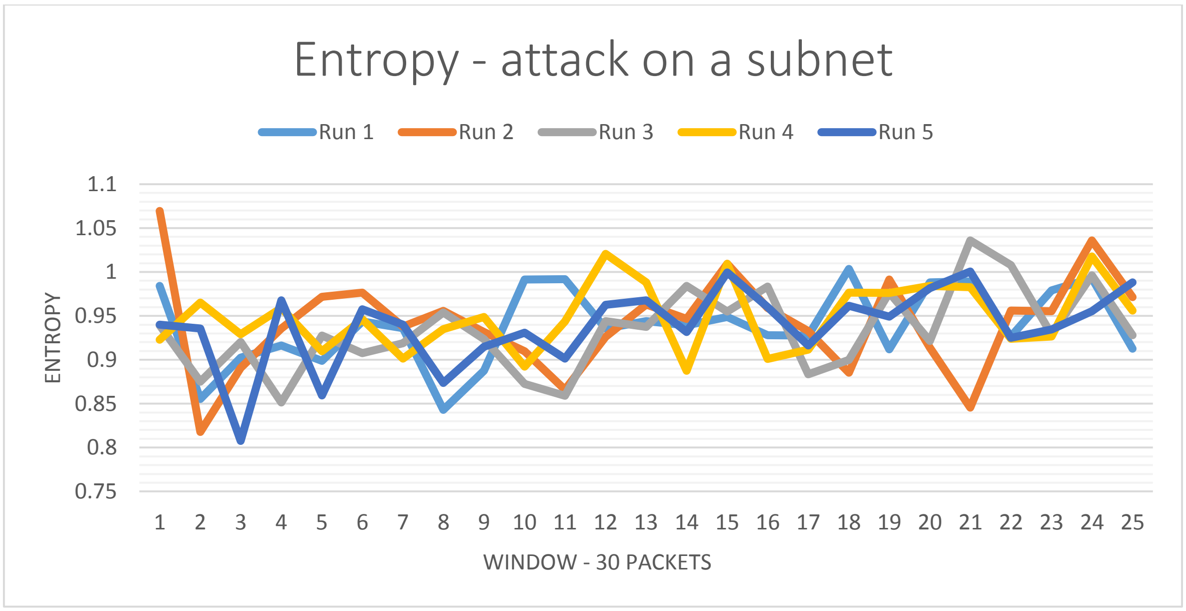


Figure 6.10 Entropy – attack traffic on a subnet, interval 0.03 seconds, test case 2

The average value for each run in 25 windows of 30 packets is presented in Table 6.9. According to it, we can conclude the average value for the entropy on a subnet of 6 hosts does not has closer value to the attack traffic on one host but with highest interval of 0.05 seconds as in the first test case. As expected, the entropy on a subnet has higher value than the entropy on one host with the same attack traffic interval of 0.03 seconds.

Average Entropy value for network under attack on 6 hosts					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	0.984240674	1.06943442	0.93918426	0.923032591	0.94018634
window no. 2	0.855251799	0.817771835	0.874900589	0.965174091	0.935526465
window no. 3	0.902544048	0.890729254	0.920376879	0.928953756	0.807539674
window no. 4	0.916459882	0.935526465	0.851174128	0.959092252	0.967669128
window no. 5	0.898654585	0.9717468	0.92769247	0.909887173	0.859331127
window no. 6	0.943683465	0.976665881	0.907463129	0.946178502	0.957599298
window no. 7	0.936108672	0.937789834	0.91895492	0.901310296	0.940025669
window no. 8	0.843017128	0.955595131	0.953941503	0.93487492	0.873666837
window no. 9	0.887394463	0.931609467	0.923614798	0.948602546	0.914806254
window no. 10	0.99139559	0.909887173	0.872405552	0.892081875	0.930866586
window no. 11	0.991815466	0.865903837	0.858909595	0.943683465	0.901310296
window no. 12	0.935526465	0.926109836	0.944103341	1.020720211	0.962750048
window no. 13	0.944103341	0.963169924	0.937530632	0.988157671	0.967669128
window no. 14	0.93918426	0.946598379	0.984240674	0.887394463	0.931448793
window no. 15	0.948602546	1.009228421	0.954943586	1.009228421	0.999649462
window no. 16	0.927951672	0.959092252	0.98323859	0.90089042	0.960094336
window no. 17	0.927531796	0.932611551	0.883477465	0.911380127	0.916459882
window no. 18	1.003307257	0.885158628	0.900049011	0.976665881	0.961516296
window no. 19	0.912003561	0.99139559	0.976665881	0.976246005	0.949022422
window no. 20	0.988157671	0.91380417	0.921378963	0.984240674	0.981584962
window no. 21	0.989159755	0.845603502	1.035869797	0.982818714	1.000392343
window no. 22	0.926529712	0.956177338	1.007967135	0.924034675	0.925036758
window no. 23	0.979160919	0.955175255	0.928533879	0.926529712	0.934686712
window no. 24	0.993076752	1.035869797	0.996314671	1.017805297	0.955175255
window no. 25	0.912542884	0.971326924	0.927951672	0.956177338	0.988157671
Average value	0.939096175	0.942159267	0.933235325	0.948606443	0.93848687
For the whole experiment	**0.940316816**				

Table 6.9 Average entropy value for network under attack on 6 hosts, attack traffic interval 0.03 seconds, test case 2

6.6.4.5 CONCLUSION REMARKS

For each experiment we launched 3750 packets in 5 runs, each of them consist of 750 packets in 25 windows. The window size is equal to 30. We collected information about the entropy in several different cases with different traffic intervals. In the next table we summarized that information of the whole experiment.

	Average entropy value	**Lowest entropy value**	**Highest entropy value**
Normal traffic – 0.1 sec	1.29	1.17	1.39
Attack on one host - 0.03 sec	0.51	0.41	0.55
Attack on one host - 0.05 sec	0.63	0.53	0.72
Attack on 6 hosts - 0.03 sec	0.94	0.8	1.03

Table 6.10 Summarized information of test case 2

6.6.5 TEST CASE 3 – DDOS ATTACK LAUNCHED VIA BOTNET OF 3 HOSTS

A growing trend in DDoS attacks is launching malicious traffic using botnets - large clusters of connected devices, in our case vehicles and/or RSUs. The impact of DDoS attacks can grow greatly if the attackers take advantage of botnet to overwhelm their victim's network. In Section 4 we explored the threats of botnets and how it works. The vast and fast development of mobile device had leads to evolution of mobile botnets, so we decided to test our two topologies under attack of botnet.

We will launch DDoS attack using botnet of 3 infected vehicles for each topology. We separated this part of the experiment from the other test cases, because we expect really low entropy values and we want to explore them individually.

This test case will be divided into 2 sub-test cases, the first experiment will be performed using the first topology (36 vehicles, 6 RSUs and one base station) and the second using the topology from test case 2 (40 hosts, 4 RSUs and one base station). The parameters of the previous experiments will remain the same. We will explore DDoS attack in 25 windows of 30 packets. This results in observation of 3750 packets. Five runs for each topology will be performed. The intensity of the attack on one host will be 0.05 seconds and the normal traffic interval will be 0.01. We will also perform tests on network under attack on a subnet of 6 hosts with attack traffic interval of 0.05 seconds. For each experiment the attack will be directed from 3 infected vehicles to one randomly chosen host. We chose that number of three hosts, because our attack will be run manually and we want to obtain more reliable results that can fit to the parameters. In real-world scenario the botnet can be even bigger.

6.6.5.1 EXPERIMENTS ON FIRST TOPOLOGY - 36 VEHICLES, 6 RSUS AND ONE BASE STATION

In the first subsection we obtained the average entropy in normal circumstances for that topology. It is 0.96. After performing the experiments on entropy of destination IP address for network under attack on one and a subnet of 6 hosts we summarized the results in the following figures and tables.

6.6.5.1.1 Attack on one host

Figure 6.11 contains information about the average entropy values for network under attack on one host with attack traffic interval 0.05 seconds. We run the attack traffic manually, and the drop in the entropy value can be observed in the first 2 windows. The lowest point of the entropy for network under attack is 0.36 and the highest is 0.67.

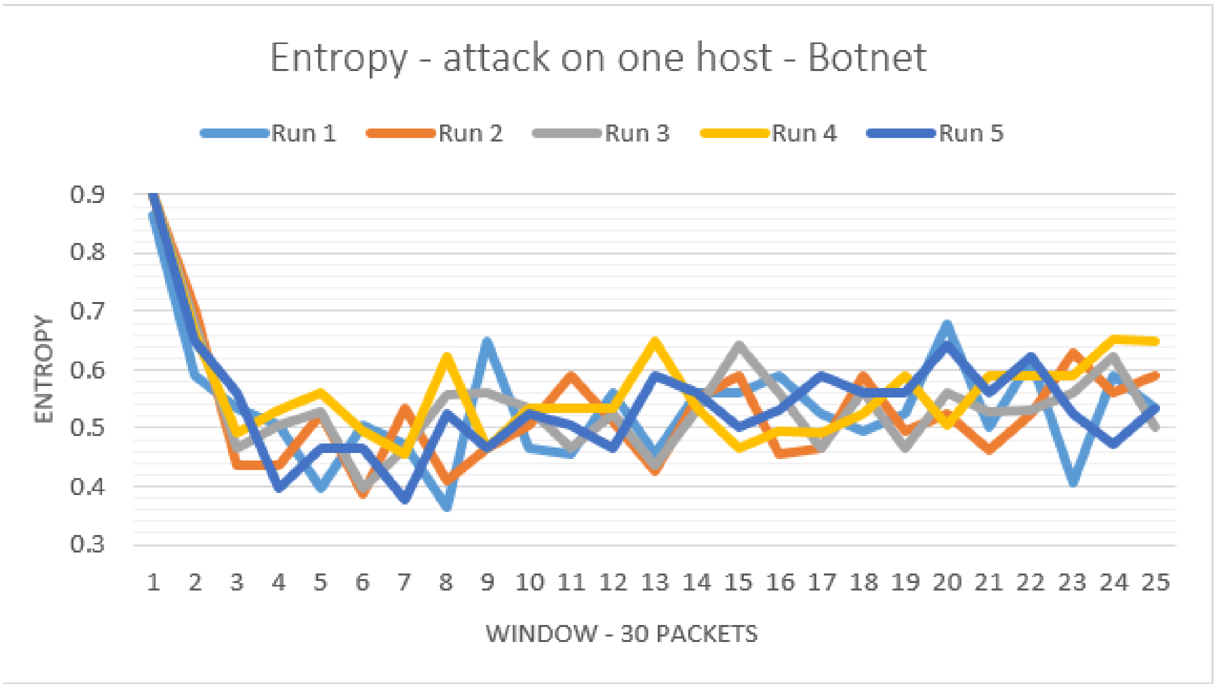


Figure 6.11 Entropy - attack on one host using botnet of 3 hosts

Next Table 6.11 gives more detailed information about the average values of the entropy for this test. We can observe huge drop comparing to the entropy in normal circumstances. The obtained average entropy is 0.54. This value decreased almost with ½ compared to the value of normal traffic for that topology.

Average Entropy value for network under attack on 1 host - Botnet					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	0.865672168	0.90231238	0.90231238	0.90231238	0.90231238
window no. 2	0.589018364	0.703700877	0.678158457	0.662561461	0.650514998
window no. 3	0.53310925	0.436074831	0.466631548	0.491545058	0.560752709
window no. 4	0.504038931	0.436074831	0.504038931	0.53168239	0.395937498
window no. 5	0.397253705	0.524107598	0.528190169	0.560752709	0.466631548
window no. 6	0.504038931	0.387360622	0.397253705	0.494275007	0.466631548
window no. 7	0.472903251	0.53310925	0.466631548	0.456143497	0.377185038
window no. 8	0.364866256	0.408431372	0.556455824	0.624006762	0.524107598
window no. 9	0.650514998	0.466631548	0.560752709	0.466631548	0.466631548
window no. 10	0.466631548	0.504038931	0.53310925	0.53310925	0.524107598
window no. 11	0.456143497	0.589018364	0.466631548	0.53310925	0.504038931
window no. 12	0.560752709	0.5120385	0.524107598	0.53310925	0.466631548
window no. 13	0.456143497	0.424897164	0.436074831	0.650514998	0.589018364
window no. 14	0.560752709	0.551751057	0.530103	0.53310925	0.560752709
window no. 15	0.560752709	0.589018364	0.644075429	0.466631548	0.50054671
window no. 16	0.589018364	0.456143497	0.560752709	0.494275007	0.53168239
window no. 17	0.524107598	0.466631548	0.466631548	0.491545058	0.589018364
window no. 18	0.494275007	0.589018364	0.563800763	0.524107598	0.560752709
window no. 19	0.524107598	0.494275007	0.466631548	0.589018364	0.560752709
window no. 20	0.678158457	0.524107598	0.560752709	0.503594764	0.644075429
window no. 21	0.50054671	0.463901599	0.528812365	0.589018364	0.560752709
window no. 22	0.616661823	0.524107598	0.53168239	0.589018364	0.624006762
window no. 23	0.406425549	0.629444248	0.560752709	0.589018364	0.524107598
window no. 24	0.590308999	0.560752709	0.624006762	0.651650221	0.472903251
window no. 25	0.53310925	0.589018364	0.50054671	0.650514998	0.53310925
Average value	0.535972475	0.530638649	0.542355886	0.564450218	0.542278476
For the whole experiment	**0.543139141**				

Table 6.11 Average entropy values for network under attack on one host using botnet of 3 hosts

6.6.5.1.2 Attack on a subnet of 6 hosts connected to one switch

Figure 6.12 gives visual information about the average entropy values for network under attack on a subnet of 6 hosts with attack traffic interval 0.05 seconds. Ignoring the results from the first two windows we can notice that the lowest point of the entropy for network under attack is 0.72 and the highest is 0.97. As expected, the results here are higher, because of the higher diversity of destination IP addresses in a subnet attack case.

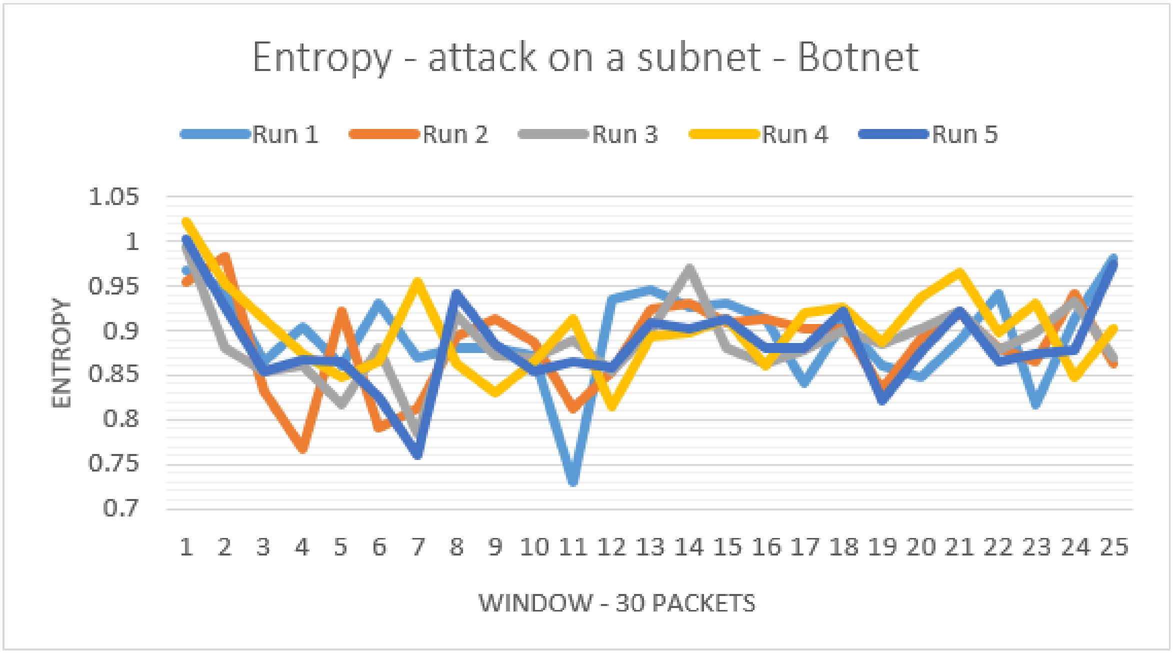


Figure 6.12 Entropy - attack on 6 hosts using botnet of 3 hosts

Table 6.12 represents the average values of the entropy for this test case. We can observe a really small drop in the entropy comparing to the value in normal circumstances. The obtained average entropy is 0.89. This minor value is only 0.07 lower than the entropy in normal traffic - 0.96. This leads us to the conclusion that using a botnet to launch an attack on a subnet will results in really hard to attack detection case. That prove the fact using the botnet, the attacker can easily overwhelm the victims' network.

Average Entropy value for network under attack on 6 hosts - Botnet					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	0.96743746	0.954943586	0.995080919	1.022724378	1.00265571
window no. 2	0.945105424	0.982587045	0.880821754	0.952939419	0.926298044
window no. 3	0.865672168	0.832107545	0.853178295	0.91380417	0.853178295
window no. 4	0.904807418	0.766750796	0.859751004	0.874668921	0.868004875
window no. 5	0.860753087	0.922381046	0.81611655	0.84867909	0.865089961
window no. 6	0.929955839	0.790548252	0.88124163	0.866092045	0.825954712
window no. 7	0.86974984	0.812038878	0.783813212	0.954943586	0.760829632
window no. 8	0.88124163	0.89273342	0.916719084	0.863668001	0.941447629
window no. 9	0.88124163	0.913384294	0.872244878	0.829612507	0.884156544
window no. 10	0.872244878	0.886160711	0.868747756	0.865672168	0.853598171
window no. 11	0.728526294	0.812458755	0.889818506	0.91380417	0.866092045
window no. 12	0.93487492	0.853178295	0.853598171	0.815696674	0.858517252
window no. 13	0.94636671	0.925036758	0.90089042	0.89273342	0.908465213
window no. 14	0.92529596	0.929955839	0.969091088	0.897652501	0.90231238
window no. 15	0.929955839	0.908465213	0.880821754	0.91380417	0.91380417
window no. 16	0.91380417	0.91380417	0.862175047	0.860753087	0.880821754
window no. 17	0.842106381	0.90231238	0.877583835	0.920376879	0.88124163
window no. 18	0.908885089	0.901310296	0.899888337	0.92529596	0.920959086
window no. 19	0.860753087	0.834531588	0.883736668	0.886160711	0.820615755
window no. 20	0.848259214	0.891311461	0.901310296	0.936528548	0.876322549
window no. 21	0.889818506	0.919957003	0.920959086	0.965433293	0.921378963
window no. 22	0.941447629	0.880821754	0.877583835	0.897393299	0.865672168
window no. 23	0.81611655	0.866092045	0.897393299	0.929955839	0.874668921
window no. 24	0.913384294	0.941447629	0.932870753	0.848259214	0.877583835
window no. 25	0.981584962	0.862175047	0.86974984	0.90231238	0.974010169
Average value	0.894375559	0.883859752	0.885807441	0.899958577	0.884947179
For the whole experiment	**0.889789702**				

Table 6.12 Average entropy values for network under attack on 6 hosts using botnet of 3 hosts

6.6.5.2 *EXPERIMENTS ON THE SECOND TOPOLOGY - 40 HOSTS, 4 RSUS AND ONE BASE STATION*

Previously we obtained the average entropy in normal circumstances for that topology. It is 1.29. After performing the tests on entropy of destination IP address for network under attack on one and a subnet of 6 hosts, with attack traffic interval 0.05 seconds, we placed the results in the following figures and tables.

6.6.5.2.1 Attack on one host

The next figure 6.13 represents the results obtained by the experiment on entropy value for network under attack on one host using botnet of 3 hosts. For the analysis of the values of the entropy we will ignore first two windows, due to the fact we launched the attack traffic manually. During the fluctuations of the entropy in this test it reached lowest point in 0.19 and highest point in 0.43.

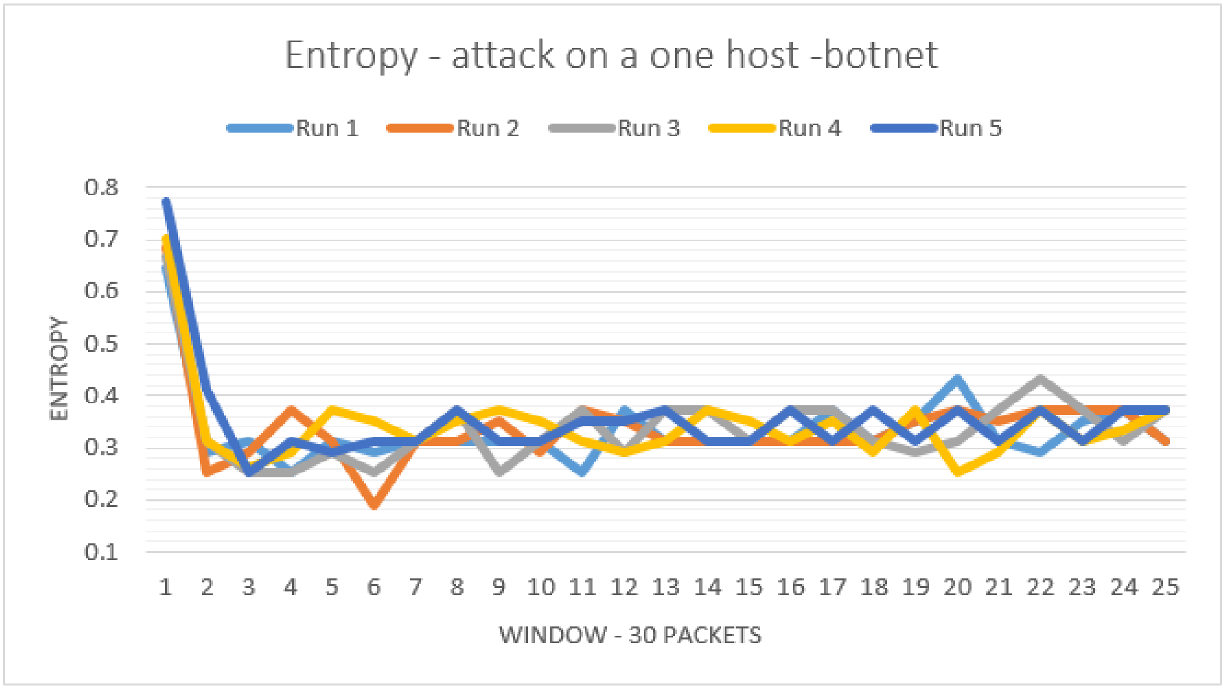


Figure 6.13 Entropy - attack on one host using botnet of 3 hosts, second topology

According to the Table 6.13 we can conclude that the average value of the entropy for attack on one host gives dramatically drop comparing to the value in normal traffic. We has been observed that phenomenon in the experiments for first topology. The average entropy value obtained by this test is 0.34. This corresponds to drop in the entropy with 0.95.

Average Entropy value for network under attack 1 host using Botnet					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	0.647175308	0.683820419	0.667243974	0.703889085	0.774717932
window no. 2	0.292102581	0.25081102	0.312171247	0.312171247	0.413061247
window no. 3	0.312171247	0.292102581	0.25081102	0.264459122	0.25081102
window no. 4	0.25081102	0.372952261	0.25081102	0.292102581	0.312171247
window no. 5	0.312171247	0.312171247	0.292102581	0.372952261	0.292102581
window no. 6	0.292102581	0.188893867	0.25081102	0.352883595	0.312171247
window no. 7	0.312171247	0.312171247	0.312171247	0.312171247	0.312171247
window no. 8	0.312171247	0.312171247	0.372952261	0.352883595	0.372952261
window no. 9	0.312171247	0.352883595	0.25081102	0.372952261	0.312171247
window no. 10	0.312171247	0.292102581	0.312171247	0.352883595	0.312171247
window no. 11	0.25081102	0.372952261	0.372952261	0.312171247	0.352883595
window no. 12	0.372952261	0.352883595	0.292102581	0.292102581	0.352883595
window no. 13	0.312171247	0.312171247	0.372952261	0.312171247	0.372952261
window no. 14	0.312171247	0.312171247	0.372952261	0.372952261	0.312171247
window no. 15	0.312171247	0.312171247	0.312171247	0.352883595	0.312171247
window no. 16	0.312171247	0.312171247	0.372952261	0.312171247	0.372952261
window no. 17	0.372952261	0.312171247	0.372952261	0.352883595	0.312171247
window no. 18	0.312171247	0.312171247	0.312171247	0.292102581	0.372952261
window no. 19	0.352883595	0.352883595	0.292102581	0.372952261	0.312171247
window no. 20	0.433129914	0.372952261	0.312171247	0.25081102	0.372952261
window no. 21	0.312171247	0.352883595	0.372952261	0.292102581	0.312171247
window no. 22	0.292102581	0.372952261	0.433129914	0.372952261	0.372952261
window no. 23	0.352883595	0.372952261	0.372952261	0.312171247	0.312171247
window no. 24	0.372952261	0.372952261	0.312171247	0.332814929	0.372952261
window no. 25	0.312171247	0.312171247	0.372952261	0.372952261	0.372952261
Average value	0.333643408	0.339147636	0.340867792	0.34382174	0.354158471
For the whole experiment	**0.342327809**				

Table 6.13 Average entropy values for network under attack on one host using botnet of 3 hosts, second topology

6.6.5.2.2 Attack on a subnet of 6 hosts

The following figure 6.14 and Table 6.14 contain information about the average entropy values for network under attack on a subnet of 6 hosts with attack traffic interval 0.05 seconds.

According to the next graph it can be noted that the lowest entropy value is 0.72 and highest is 0.92. Comparing to the results of the entropy for network under attack on one host, here the entropy experienced slight decrease.

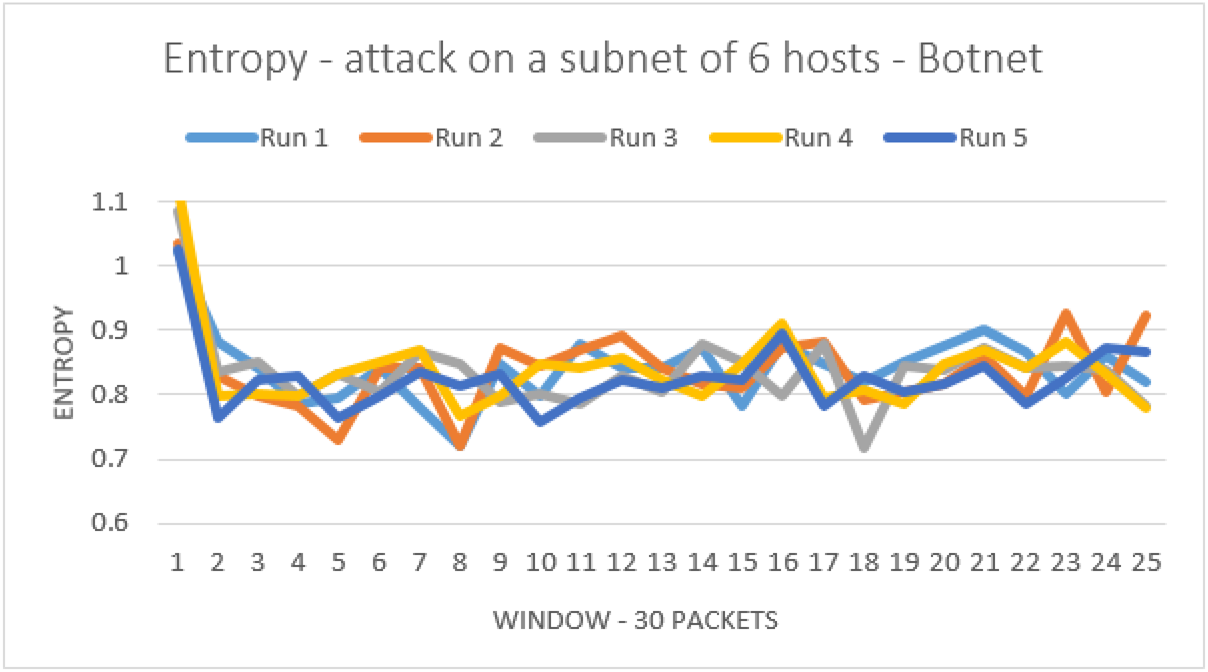


Figure 6.14 Entropy - attack on 6 hosts using botnet of 3 hosts, second topology

Table 6.14 contains information about the average values of the entropy. We can observe drop in the entropy comparing to the value in normal circumstances. The obtained average entropy is 0.84. This value is 0.45 lower than the entropy in normal traffic – 1.29. Comparing to the results of average entropy for the same case, but for the previous topology, here the decrease is not that notable.

Average Entropy value for network under attack on 6 hosts					
Entropy	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
window no. 1	1.022956047	1.03687188	1.084584006	1.124721338	1.02538009
window no. 2	0.882243714	0.829683502	0.836185216	0.799125128	0.764487427
window no. 3	0.842757926	0.799125128	0.850172044	0.802040042	0.823530669
window no. 4	0.784395419	0.784163751	0.799964881	0.797309169	0.829871709
window no. 5	0.794625924	0.72934017	0.832107545	0.831687668	0.762904793
window no. 6	0.842757926	0.843599335	0.803622676	0.850172044	0.796889293
window no. 7	0.778474255	0.843599335	0.865903837	0.871662671	0.83576534
window no. 8	0.721533708	0.720272423	0.84725713	0.766982465	0.814855264
window no. 9	0.848259214	0.874480713	0.788473091	0.797049967	0.833529505
window no. 10	0.799125128	0.846023378	0.801969048	0.84867909	0.75817392
window no. 11	0.87981967	0.871242794	0.785397503	0.840684421	0.794625924
window no. 12	0.843017128	0.891731337	0.829871709	0.858489718	0.823299
window no. 13	0.842338049	0.840684421	0.805697838	0.823530669	0.81180721
window no. 14	0.872405552	0.815696674	0.878978261	0.798703595	0.829871709
window no. 15	0.781712174	0.812038878	0.852596088	0.848259214	0.824693426
window no. 16	0.87981967	0.872405552	0.797703168	0.909887173	0.893735504
window no. 17	0.84725713	0.882734584	0.87981967	0.795628007	0.781971376
window no. 18	0.819031464	0.793651374	0.718688132	0.808541757	0.830453916
window no. 19	0.852596088	0.804464086	0.846023378	0.786468924	0.804464086
window no. 20	0.875320466	0.818379919	0.838260378	0.84725713	0.818191711
window no. 21	0.900049011	0.859519335	0.874480713	0.870822918	0.846023378
window no. 22	0.86832788	0.798542921	0.842338049	0.843599335	0.785209295
window no. 23	0.80222825	0.927951672	0.845603502	0.882734584	0.826768587
window no. 24	0.859751004	0.805697838	0.839100131	0.831875876	0.874480713
window no. 25	0.819774345	0.922612715	0.782159584	0.779896215	0.866323713
Average value	0.842423086	0.840980549	0.837078303	0.840632365	0.826292302
For the whole experiment	**0.837481321**				

Table 6.14 Average entropy values for network under attack on 6 hosts using botnet of 3 hosts, second topology

6.6.5.3 CONCLUSION REMARKS

For further convenience the information about the whole test case 3 is presented in the next two tables – Table 6.15 and Table 6.16.

	Average entropy value	**Lowest entropy value**	**Highest entropy value**
Normal traffic – 0.1 sec	0.96	0.68	1.07
Attack on one host - 0.05 sec	0.54	0.36	0.67
Attack on 6 hosts - 0.05 sec	0.89	0.72	0.97

Table 6.15 Summarized information for the first topology

	Average entropy value	**Lowest entropy value**	**Highest entropy value**
Normal traffic – 0.1 sec	1.29	1.17	1.39
Attack on one host - 0.05 sec	0.34	0.19	0.43
Attack on 6 hosts - 0.05 sec	0.83	0.72	0.92

Table 6.16 Summarized information for the second topology

6.7 CHOOSING A THRESHOLD

To detect DDoS attack our solution requires threshold. In [79] the author has implemented a solution based on which if the entropy is lower than the threshold, and it persists for 5 windows in a row, an attack is in progress. We decided to decrease this 5 windows to 3 windows, because we are interested in really small time interval, due to the dynamic nature of our VANET network.

To discover the scope for an optimal threshold in the previous subsections we have been done sequence of experiments to examine how the attack affects the entropy. The tests cover an attack to one host and a subnet of 6 hosts. To compare various rates of incoming packets, we varied the rate of normal and attack traffic to increase and decrease the intensity of DDoS on the controller. According to Table 6.5, Table 6.10, Table 6.15 and Table 6.16 we can choose a threshold for each of the experiments. To choose a threshold for each topology, we also summarized the information collected by the experiments in the following graphs - figure 6.15, figure 6.16., figure 6.17 and figure 6.18.

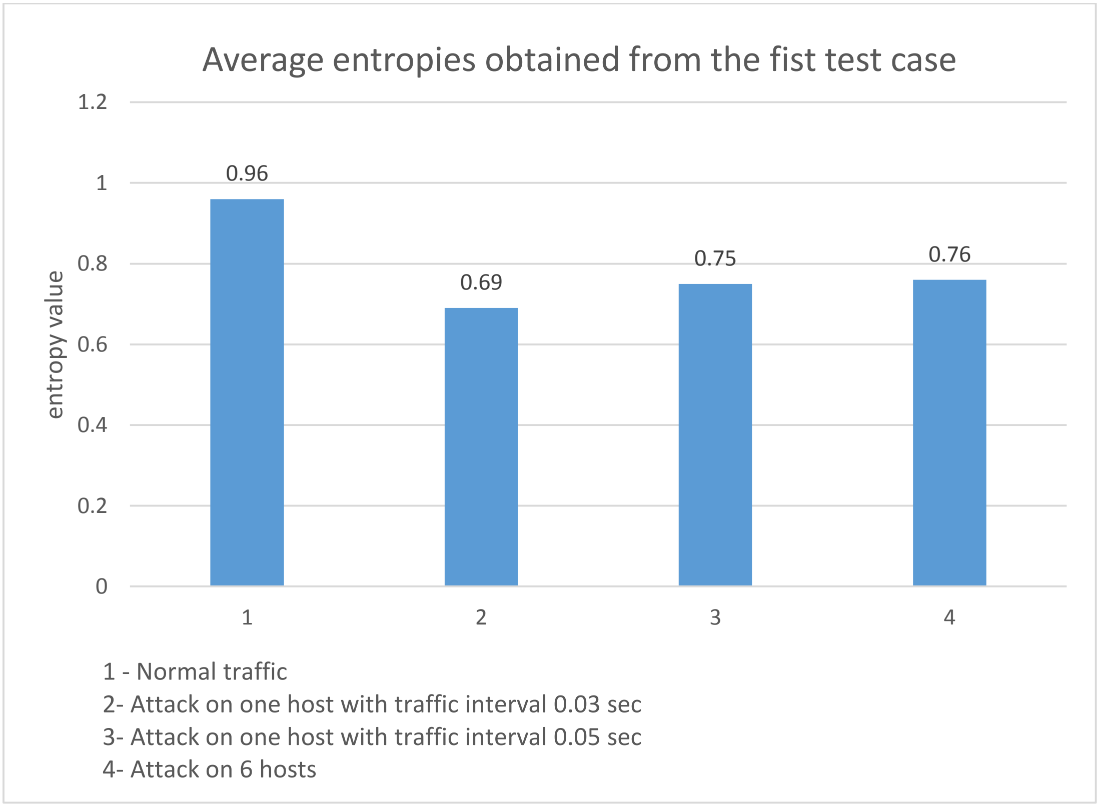


Figure 6.15 Average entropies obtained from the first test case

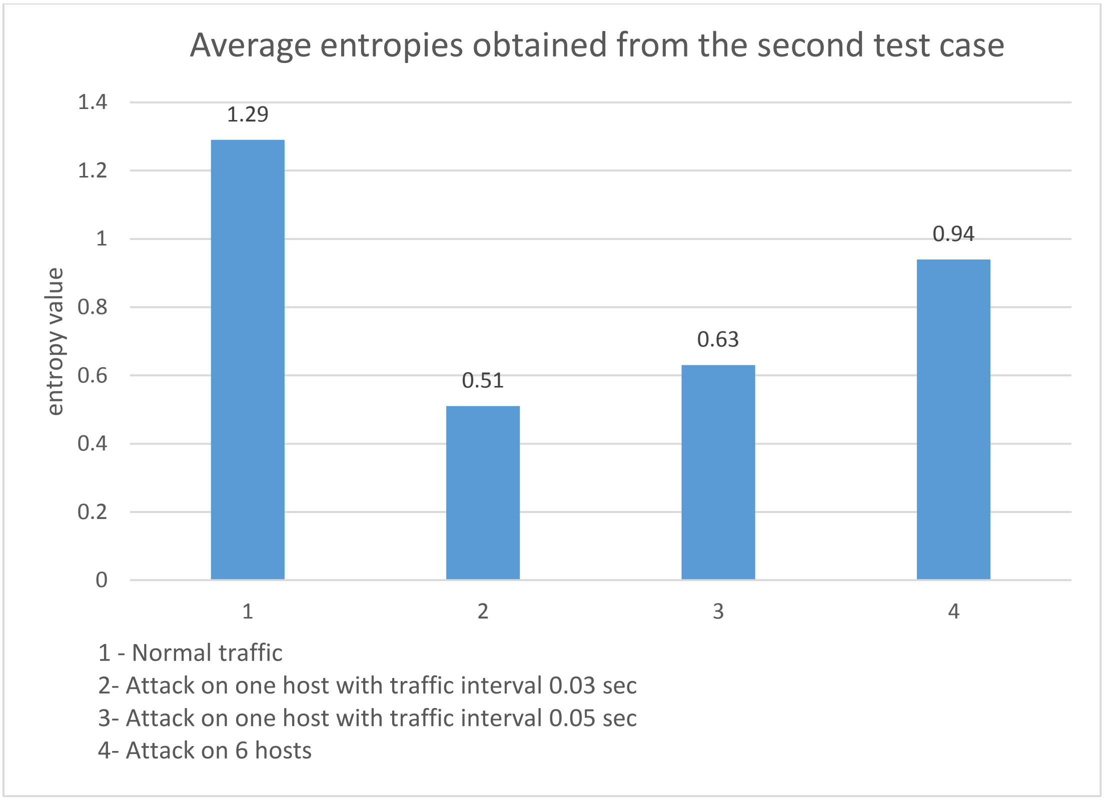


Figure 6.16 Average entropies obtained from the second test case

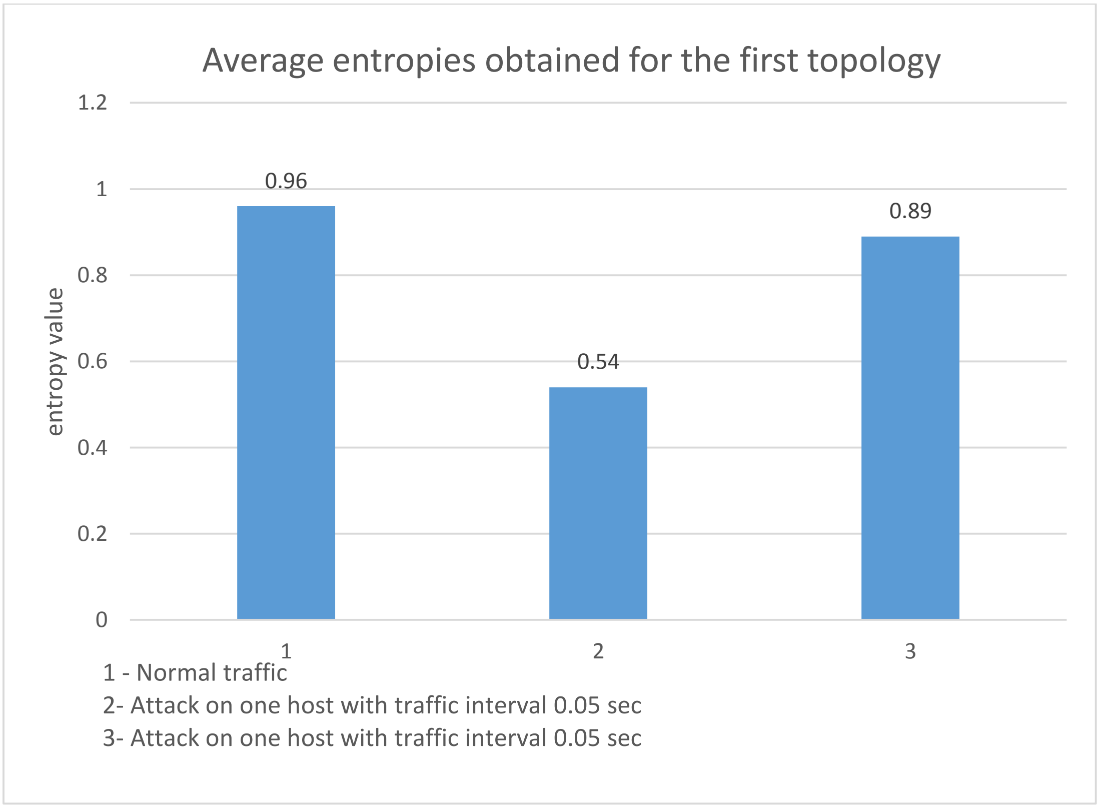


Figure 6.17 Average entropies obtained for the first topology

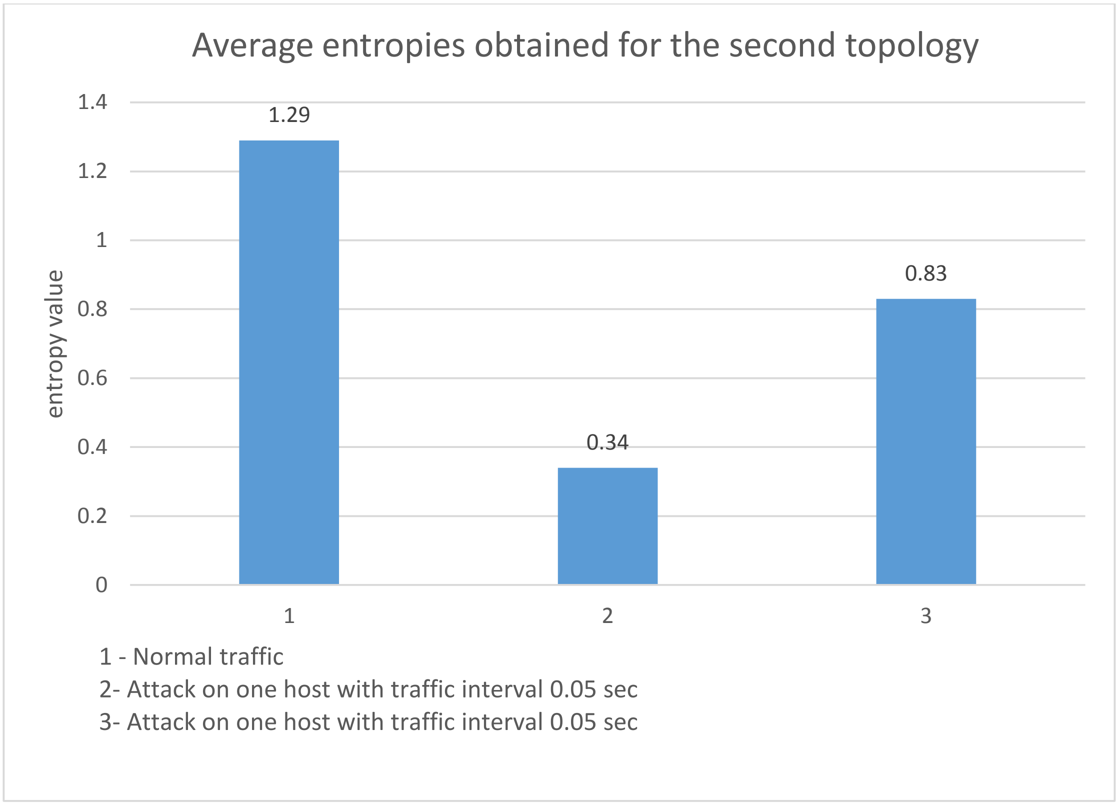


Figure 6.18 Average entropies obtained for the second topology

6.7.1.1 THRESHOLD FOR THE FIRST EXPERIMENT

In the first test case the lowest average entropy reached 0.69 when the network was under attack with attack traffic interval equal to 0.03 seconds. The results of the attack on one host with interval 0.05 seconds and of the attack on subnet of 6 hosts with interval of 0.03 are closer, respectively 0.75 and 0.76. The entropy in normal circumstances is 0.96. The difference between the entropy of normal traffic and the highest entropy of attack traffic is 0.2. For the purpose of our work, we are choosing the threshold to be 0.76. Only then we can be sure that our solution will detect the attack if it occurs and the entropy is lower than 0.76 and it lasts for at least 3 windows.

6.7.1.2 THRESHOLD FOR THE SECOND EXPERIMENT

Comparing to the first experiment here the entropy of destination IP addresses for normal traffic is higher – 1.29. The reason of that is we didn`t change the window size and it remained the same – 30 packets. When we included more hosts to our network, the diversity between the IP addresses increased. The code for packet generation can choose to send the traffic between more nodes in the same window size. The lowest entropy for the second scenario was reached when the network was under attack on one host with traffic interval 0.03 seconds. The entropy value is 0.51. As expected, the highest entropy of attack traffic was reached when 6 hosts were under attack – 0.94. We can notice that value is really close to the value of the entropy for normal traffic in the first test case. Therefore, we can conclude, it is very important to find and adjust the threshold according to the current network topology. That was one of the goals of our experiments, to show how in dynamic network topology the parameters has to be carefully considered, because constant values are not applicable for VANET scenario. According to that we are choosing the threshold for the second experiment to be 0.94. That corresponds to drop in the entropy with 0.35.

6.7.1.3 THRESHOLD FOR THE THIRD EXPERIMENT

In the last part of our work we performed tests on the entropy of destination IP addresses using botnets. According to tables 6.15 and table 6.16 and figure s 6.17 and figure 6.18 we can choose an appropriate threshold for each topology.

For the first topology we obtained lowest average entropy 0.54, when the network was under attack with attack traffic directed to one host and with attack traffic interval equal to 0.05 seconds. The results of attack on a subnet of 6 hosts with interval 0.05 seconds was 0.89. This corresponds to a really slight drop in the entropy – the decrease is with only 0.07. We proved that the attacker can benefit of botnets and overwhelm the network without changing dramatically some traffic features. Anyway, we can adjust the threshold to be 0.89, if we are aware of the nature of the attack, i.e. if we know that attempts of DDoS attacks launched by a botnet can occur in the network. In our test case that is not reliable, because if we assume the threshold to be 0.89 some incorrect detections can occur, due to the fact the average entropy for attack on one host that we obtained was 0.54. We can choose 0.89, but we have to increase the number of windows, that we will take into account, when the controller are making the decision about the occurrence of DDoS attack. This is also not really reliable in our case, because as we mentioned before, we have been chosen 3 windows in a row, due to the fact we are interested in really small time interval.

For the second topology the situation with choosing entropy is almost the same. It is really risky to choose a threshold for that test case, because we obtained really different values between both tests. For attack on one host we observed average entropy equal to 0.34 and for attack on a subnet of 6 hosts the value is 0.83. The average entropy in normal circumstances is 1.29. The difference between the lowest entropy and this in normal traffic environment is really huge – 0.95

comparing to the difference (0.46) between the entropy of attack on a subnet and the same value of 1.29. We can choose the threshold to be 0.83 by taking into account the same assumptions that we made for the first topology.

7 CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

Since the security in the networking, can be consider as a major concern, protecting the operating system of SDN-based VANET architecture by detecting a DDoS attack was the centre of this diploma thesis. Early detection of the occurrence of attack was crucial point in our architecture, due to the fact that if the SDN controller was compromised, the whole network could become unreachable. Furthermore, for our VANET test scenario, which includes applications that are working with life critical information, the importance of security is even more significant.

We conducted a research on different methods for detecting a DDoS attack proposed by the literature and we have noticed that each of them works differently. In our thesis, we directed our efforts to a solution, which works extremely good for SDN scenario, based on its specifications, standards, points of strength and limitations. We wanted to implement it in SDN-based VANET architecture and to check if it will work properly with different topologies in order to provide a basis of a solution for DDoS attack detection in dynamic environments. We made a precise definition of each parameter that we used. We decided to test the solution in three different test cases, using two SDN-based VANET topologies. Each test case covers several experiments on different scenarios, namely attack on host, attack on a subnet of 6 hosts and attack launched by usage of botnet. The test included launching normal and DDoS attack traffic with spoofed source IP addresses. Based on traffic features, such as destination IP addresses and following a sequence of 3 windows of 30 packet, the entropy was used to measure the degree of randomness of the packets that are received by the SDN controller. By computing the entropy and after appropriate definition of the threshold for each case, we were capable to detect an attack on one host or a subnet of 6 hosts and when the attack is launched by botnet of 3 hosts. Definition of different thresholds was needed for each test case, since we examined two different topologies that simulate dynamic VANET scenario. We believe that based on our test results, a dynamic model, which adjust its parameters instantly after obtaining current status from the network, can be developed. The SDN controller can take into account the number of switches and hosts that are connected to it and according to that current topology, to change its threshold. We observed that the solution works really smoothly for VANET network, but we have to admit that in the test case of launching DDoS attack using the latest trend in network attacking, namely botnet, choosing of appropriate threshold is not that easy, as in the other test cases. Anyway, with some assumptions, we can adequately detect an attack. As another conclusion remark, we can mention that the proposed solution is not only efficient in detection, but it also requires minimal code addition to the controller module. Therefore, we suppose that our method will not influence dramatically the usage of CPU.

7.2 FUTURE WORK

As we simulate our solution to obtain results that can be taken as a basis for development of a dynamic DDoS attack detection algorithm and to implement it in the SDN controller, we consider that as the main direction of our future work.

The next step is mitigation of DDoS attacks in SDN-based VANET scenario. The first part of mitigation will be the detection of the source of the attack and after that its removal from the network.

The testing and implementation of our detection algorithm for larger, dynamic topologies to see its performance is another direction for future research.

Since our work was focused on the control plane communication, i.e. in our case the communication between the vehicles/RSUs and the controller, potential objective for future work will be the examination of the data plane communication.

Last direction of investigation could be the examination of how different functions in the controller related to detection and mitigation of DDoS attacks can influence the usage of CPU of the controller.

Appendix A: Starting Mininet command

First topology:

```
sudo mn --switch ovsk  --topo tree, depth = 2, fanout = 6 –controller = remote,ip = 127.0.0.1, port= 6633
```

Second topology:

```
sudo mn --switch ovsk --topo single,44 --controller=remote,ip=127.0.0.1,port=6633
```

Appendix B: Functions added to the controller to compute the entropy values

```
count = 0                    # preserving count for a window 30
entDic = {}                # hash table for (IP, number of time it shown in a window)
ipList = []                # IP address list
dstEnt = []                # list of entropies

def statcolect(self, element):
      # as an input the function obtains the destination IP address of each packet
    l=0
    self.count +=1
    self.ipList.append(element)
      # we reached 30 fill hash table
    if self.count ==30:
       for i in self.ipList:
         l +=1
         if i not in self.entDic:
            self.entDic[i]=0
         self.entDic[i] +=1
       self.entropy(self.entDic)
       print 'Number of time each IP appeard in this window = ', self.entDic
       self.entDic={}
       self.ipList=[]
       l=0
       self.count=0

def entropy(self, lists):
     # with this function the Entropy is computed
```

```
print "The computation of the entropy started!"
l=float(30)
elist=[]
print 'Number of time = ', lists.values()
for p in lists.values():
  p = float(p)
  c = float(p/l)
  #print 'Division =',c
  elist.append(-c*math.log10(c))
print 'The entropy has been calculated! Entropy = ',sum(elist)
self.dstEnt.append(sum(elist))
if (len(self.dstEnt))==25:
   print '750 packets reached! The full run is: ', self.dstEnt
   self.dstEnt={}
```

Appendix C: Starting the POX controller

```
./pox.py forwarding.l3_learning1
```

Appendix D: Code that generates normal traffic between the hosts

```
#!/usr/bin/env python
import sys
import getopt
import time
from os import popen
from scapy.all import sendp, IP, UDP, Ether, TCP
from random import randrange

def sourceIPgen():

    not_valid=[10,127,254,255,1,2,169,172,192]
   first=randrange(1,256)
    while first in not_valid:
      first=randrange(1,256)
  ip=".".join([str(first),str(randrange(1,256)),
              str(randrange(1,256)),str(randrange(1,256))])
  return ip
def gendest(start, end):
       first=10
       second=0; third = 0;
       ip = ".".join([str(first),str(second),
                    str(third),str(randrange(start,end))])
      return ip
def main():
  print("Normal traffic generator: ")
  try:
    opts, args=getopt.getopt(sys.argv[1:],'s:e:',['start=','end='])
  except getopt.GetoptError:
    sys.exit(2)
  for opt, arg in opts:
    if opt=='-s':
       start=int(arg)
       print("Start: "+str(start))
    elif opt=='-e':
       end=int(arg)
  if start=='':
    sys.exit()
  if end=='':
    sys.exit()

      print(str(start)+" "+str(end))
  interface = popen('ifconfig | awk \'/eth0/ {print $1}\'').read()
  print("Interface: "+str(interface))
  for i in xrange(1000):
      packets = Ether()/IP(dst=gendest(start, end),src=sourceIPgen())/UDP(dport=80,sport=2)
      print(repr(packets))
      sendp(packets,iface=interface.rstrip(),inter=0.1)

#main
if __name__=="__main__":
  main()
```

Appendix E: Code that generates DDoS attack traffic on one host.

```
#!/usr/bin/env python

import sys

import time

from os import popen

from scapy.all import sendp, IP, UDP, Ether, TCP

from random import randrange


def sourceIPgen():

  not_valid=[10,127,254,255,1,2,169,172,192]

  first=randrange(1,256)

  while first in not_valid:

   first=randrange(1,256)

   print ("purtvi print"+str(first))

  ip=".".join([str(first),str(randrange(1,256)),

           str(randrange(1,256)),str(randrange(1,256))])

  print ip

  return ip


def main():

  dstIP=sys.argv[1:]

  print dstIP

  src_port=80

  dst_port=1
  interface=popen('ifconfig | awk \'/eth0/ {print $1}\'').read()
  for i in xrange(0,500):
    packets=Ether()/IP(dst=dstIP,src=sourceIPgen())/UDP(dport=dst_port,sport=src_port)
    print(repr(packets))
    sendp(packets,iface=interface.rstrip(),inter=0.03)

#main
if __name__=="__main__":
  main()
```

Appendix F: Code that generates DDoS attack traffic on 6 hosts.

```
#!/usr/bin/env python

import sys
import time
import getopt
from os import popen
from scapy.all import sendp, IP, UDP, Ether, TCP
from random import randrange

def sourceIPgen():
  not_valid=[10,127,254,255,1,2,169,172,192]
  first=randrange(1,256)
  while first in not_valid:
   first=randrange(1,256)
   print ("purtvi print"+str(first))
  ip=".".join([str(first),str(randrange(1,256)),
          str(randrange(1,256)),str(randrange(1,256))])
  print ip
  return ip

def gendest(start,end):
  first = 10
  second = 0;third = 0;
  ip = ".".join([str(first),str(second),str(third),
            str(randrange(start,end))])
  return ip
```

```
def main():
  try:
    opts, args = getopt.getopt(sys.argv[1:],'s:e:',['start=','end='])
  except getopt.GetoptError:
    sys.exit(2)
  for opt, arg in opts:
    if opt =='-s':
        start = int(arg)
    elif opt == '-e':
        end = int(arg)
  if start =='':
     sys.exit()
  if end =='':
    sys.exit()
 interface=popen('ifconfig | awk \'/eth0/ {print $1}\'').read()
 for i in xrange(0,500):
   packets=Ether()/IP(dst=gendest(start, end),src=sourceIPgen())/UDP(dport=80,sport=1)
   print(repr(packets))
   sendp(packets,iface=interface.rstrip(),inter=0.03)

#main
if __name__=="__main__":
 main()
```

REFERENCES

[1] Open Networking Foundation (ONF). [Online]. Available: https://www. opennetworking.org/

[2] "Software-defined networking: The new norm for networks," Palo Alto, CA, USA, White Paper, Apr. 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdnnewnorm.pdf

[3] "A Survey on Software-Defined Networking," Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, Haiyong Xie, Communications Surveys & Tutorials, IEEE (2015)

[4] J. Smith et al., "Switchware: Accelerating network evolution," CIS Dept., Univ. Pennsylvania, Philadelphia, PA, USA, White Paper, 1996.

[5] D. Alexander et al., "The SwitchWare active network architecture," IEEE Netw., vol. 12, no. 3, pp. 29–36, May/Jun. 1998.

[6] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek, "The click modular router," ACM Trans. Comput. Syst., vol. 18, no. 3, pp. 263–297, Aug. 2000.

[7] M. Handley, O. Hodson, and E. Kohler, "XORP: An open platform for network research," ACM SIGCOMM Comput. Commun. Rev., vol. 33, no. 1, pp. 53–57, Jan. 2003.

[8] Quagga Routing Software Suite. [Online]. Available: http://www.nongnu.org/quagga/

[9] The BIRD Internet Routing Daemon. [Online]. Available: http://bird. network.cz/

[10] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in Proc. ACM SIGCOMM Workshop FDNA, 2004, pp. 5–12.

[11] L. Yang, R. Dantu, T. Anderson, and R. Gopal, Forwarding and Control Element Separation (ForCES) Framework, Apr. 2004, RFC 3746. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[12] J. Rexford et al., "Network-wide decision making: Toward a wafer-thin control plane," in Proc. HotNets, 2004, pp. 59–64.

[13] M. Casado et al., “SANE: A protection architecture for enterprise networks,” in Proc. 15th Conf. USENIX-SS, Berkeley, CA, USA, 2006, vol. 15, p. 10.

[14] S. Shenker, M. Casado, T. Koponen, and N. McKeown, “The future of networking, and the past of protocols,” presented at the Open Networking Summit, Stanford, CA, USA, 2011. [Online]. Available: http://www.opennetsummit.org/archives/apr12/site/talks/shenker-tue.pdf

[15] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, “Toward software-defined middlebox networking,” in Proc. 11th ACM Workshop Hot Topics Netw., 2012, pp. 7–12.

[16] T. Koponen et al., “Architecting for innovation,” ACM SIGCOMM Comput. Commun. Rev., vol. 41, no. 3, pp. 24–36, Jul. 2011.

[17] A. Sharafat, S. Das, G. Parulkar, and N. McKeown, “MPLS-TE and MPLS VPNS with OpenFlow,” ACM SIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 452–453, Aug. 2011.

[18] N. Blefari-Melazzi, A. Detti et al., “An OpenFlow-based testbed for information centric networking,” in Proc. Future Netw. Mobile Summit, 2012, pp. 4–6.

[19] H. Yin et al., SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains, Jun. 2012, Internet draft. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[20] Open Network Foundation “OpenFlow Switch Specification” Version 1.4.0 (Wire Protocol 0x05) October 14, 2013

[21] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, Guru Parulkar, ”FlowVisor: A Network Virtualization Layer”, 2009

[22] Dmitry A. Drutskoy, “Software-Defined Network Virtualization with FlowN”, Master’s Thesis, June 2012

[23] Zdravko Bozakov, Panagiotis Papadimitriou “AutoSlice: Automated and Scalable Slicing for Software-Defined Networks”, Institute of Communications Technology

[24] Ali Al-Shabibi, Marc De Leenheer, Ayaka Koshibe, Guru Parulkar and Bill Snow - Open Networking Laboratory, Matteo Gerola and Elio Salvador - CREATE-NET “OpenVirteX: Make Your Virtual SDNs Programmable”, 2014

[25] R. Alimi, R. Penno, and Y. Yang, ALTO Protocol, Feb. 2013, Internet Draft. [Online]. Available: http://tools.ietf.org/id/ draft-ietf-alto-protocol-14.txt

[26] V. Gurbani, M. Scharf, T. Lakshman, V. Hilt, and E. Marocco, “Abstracting network state in Software Defined Networks (SDN) for rendezvous services,” in Proc. IEEE ICC, 2012, pp. 6627–6632.
[27] E. Kissel, G. Fernandes, M. Jaffee, M. Swany, and M. Zhang, “Driving software defined networks with XSP,” in Proc. Workshop SDN/IEEE Int. Conf. Commun., 2012, pp. 6616–6621
[28] E. Keller and J. Rexford, “The “Platform as a service” model for networking,” in Proc. INM/WREN, 2010, p. 4.
[29] J. Fu, P. Sjödin, and G. Karlsson, “Intra-domain routing convergence with centralized control,” Comput. Netw., vol. 53, no. 18, pp. 2985–2996, Dec. 2009.
[30] K. K. Lakshminarayanan, I. Stoica, S. Shenker, and J. Rexford, “Routing as a service,” EECS Dept., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2006-19, Feb. 2006.
[31] R. Wang, D. Butnariu, and J. Rexford, “OpenFlow-based server load balancing gone wild,” in Proc. 11th USENIX Conf. Hot-ICE Netw. Serv., 2011, p. 12.
[32] P. Pisa et al., “OpenFlow and Xen-based virtual network migration,” in Communications: Wireless in Developing Countries and Networks of the Future. Berlin, Germany: Springer-Verlag, 2010, pp. 170–181.
[33] M. Koerner and O. Kao, “Multiple service load-balancing with OpenFlow,” in Proc. 13th IEEE Int. Conf. HPSR, 2012, pp. 210–214.
[34] N. Handigol et al., “Plug-n-Serve: Load-balancing web traffic usingOpenFlow,” in Proc. ACM SIGCOMM Demo, Barcelona, Spain,2009. [Online]. Available: http://conferences.sigcomm.org/sigcomm/2009/demos/sigcomm-pd-2009-final26.pdf
[35] A. Ferguson, A. Guha, J. Place, R. Fonseca, and S. Krishnamurthi, “Participatory networking,” in Proc. Hot-ICE, San Jose, CA, USA, 2012, p. 2.
[36] K. Jeong, J. Kim, and Y. Kim, “QoS-aware network operating system for software defined networking with generalized OpenFlows,” in Proc. IEEE NOMS, 2012, pp. 1167–1174.
[37] G. Wang, T. E. Ng, and A. Shaikh, “Programming your network at runtime for big data applications,” in Proc. 1st Workshop HotSDN, 2012, pp. 103–108.
[38] K.-K. Yap et al., “Blueprint for introducing innovation into wireless mobile networks,” in Proc. 2nd ACM SIGCOMM Workshop VISA, 2010, pp. 25–32.
[39] K.-K. Yap et al., “OpenRoads: Empowering research in mobile networks,” SIGCOMM Comput. Commun. Rev., vol. 40, no. 1, pp. 125–126, Jan. 2010.

[40] K. Yap, S. Katti, G. Parulkar, and N. McKeown, “Delivering capacity for the mobile internet by stitching together networks,” in Proc. ACM Workshop Wireless Students, 2010, pp. 41–44.

[41] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, “Towards programmable enterprise WLANS with Odin,” in Proc. 1st Workshop HotSDN, 2012, pp. 115–120.

[42] Z. Kerravala, “As the value of enterprise networks escalates, so does the need for configuration management,” Enterprise Computing & Networking, The Yankee Group Report, Boston, MA, USA, 2004.

[43] N. Handigol, B. Heller, V. Jeyakumar, D. Maziéres, and N. McKeown, “Where is the debugger for my software-defined network?” in Proc. 1st Workshop HotSDN, 2012, pp. 55–60.

[44] A.Wundsam, D. Levin, S. Seetharaman, and A. Feldmann, “OFRewind: Enabling record and replay troubleshooting for networks,” in Proc. USENIX Annu. Tech. Conf., 2011, p. 29.

[45] J. Fu, P. Sjödin, and G. Karlsson, “Intra-domain routing convergence with centralized control,” Comput. Netw. , vol. 53, no. 18, pp. 2985–2996, Dec. 2009.

[46] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, “Enabling fast failure recovery in OpenFlow networks,” in Proc. 8th Int. Workshop DRCN, 2011, pp. 164–171.

[47] G. Lu, R. Miao, Y. Xiong, and C. Guo, “Using CPU as a traffic coprocessing unit in commodity switches,” in Proc. 1stWorkshop HotSDN, 2012, pp. 31–36.

[48] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” in Proc. 35th IEEE Conf. LCN, 2010, pp. 408–415.

[49] G. Gibb, H. Zeng, and N. McKeown, “Outsourcing network functionality,” in Proc. 1st Workshop HotSDN, 2012, pp. 73–78.

[50] G. Huang, C. Chuah, S. Raza, and S. Seetharaman, “Dynamic measurement-aware routing in practice,” IEEE Netw., vol. 25, no. 3, pp. 29–34, May/Jun. 2011.

[51] A. Ferguson, A. Guha, J. Place, R. Fonseca, and S. Krishnamurthi, “Participatory networking,” in Proc. Hot-ICE, San Jose, CA, USA, 2012, p. 2.

[52] J. H. Jafarian, E. Al-Shaer, and Q. Duan, “OpenFlow random host mutation: Transparent moving target defense using software defined networking,” in Proc. 1st Workshop HotSDN, 2012, pp. 127–132.

[53] M. Mendonca, S. Seetharaman, and K. Obraczka, “A flexible in-network IP anonymization service,” in Proc. IEEE ICC, 2012, pp. 6651–6656.

[54]A. Bianzino, C. Chaudet, D. Rossi, and J. Rougier, “A survey of green networking research,” IEEE Commun. Surveys Tuts., vol. 14, no. 1, pp. 3–20, Dec. 2012.
[55] B. Heller et al., “ElasticTree: Saving energy in data center networks,” in Proc. 7th USENIX Conf. NSDI, 2010, p. 17.
[56] N. Chowdhury and R. Boutaba, “Network virtualization: State of the art and research challenges,” IEEE Commun. Mag., vol. 47, no. 7, pp. 20–26, Jul. 2009.
[57] D. Turull, M. Hidell, and P. Sjodin, “Using libNetVirt to control the virtual network,” in Proc. IEEE Int. Conf. CLOUDNET, 2012, pp. 148–152.
[58] S. Gutz, A. Story, C. Schlesinger, and N. Foster, “Splendid isolation: A slice abstraction for software-defined networks,” in Proc. 1st Workshop HotSDN, 2012, pp. 79–84.
[59] T. Benson, A. Akella, A. Shaikh, and S. Sahu, “CloudNaaS: A cloud networking platform for enterprise applications,” in Proc. 2nd ACM SOCC, 2011, pp. 8:1–8:13.
[60] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, “Applying NOX to the datacenter,” in Proc. HotNets, New York, NY, USA, 2009. [Online]. Available: http://conferences.sigcomm.org/hotnets/ 2009/papers/hotnets2009-final103.pdf
[61] B. Pfaff et al., “Extending networking into the virtualization layer,” in Proc. HotNets, New York, NY, USA, 2009. [Online].
Available: http://conferences.sigcomm.org/hotnets/2009/papers/ hotnets2009-final143.pdf
[62] M. Moshref, M. Yu, A. Sharma, and R. Govindan, “Scalable rule management for data centers,” in Proc. 10th USENIX Conf. NSDI, 2013, pp. 157–170.
[63] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: State-of-the-art and research challenges,” J. Internet Serv. Appl., vol. 1, no. 1, pp. 7–18, May 2010.
[64] OpenDaylight. [Online]. Available: http://www.opendaylight.org/
[65] V. Mann, A. Vishnoi, K. Kannan, and S. Kalyanaraman, “CrossRoads:
Seamless VM mobility across data centers through software defined networking,” in Proc. IEEE NOMS, 2012, pp. 88–96.
[66] Imrich Chlamtac, Marco Conti, and Jennifer J Liu. Mobile Ad Hoc Networking: Imperatives and Challenges. Ad Hoc Networks, 1(1):13–64, Jul 2003.
[67] Stephan Eichler. Security Challenges in MANET-based Telematics Environments. In Proceedings of the 10th Open European Summer School and IFIP WG 6.3 Workshop,
Jun 2004.

[68] Jeroen Hoebeke, Ingrid Moerman, Bart Dhoedt, and Piet Demeester. An Overview of Mobile Ad Hoc Networks: Applications and Challenges. The Communications Network, 3(3), 2004

[69] "FCC Report and Order 03-324: Amendment of the Commission's Rules Regarding Dedicated Short-Range Communication Services in the 5.850-5.925 GHz Band," December 17, 2003.

[70] V. Rai, F. Bai, J. Kenney and K. Laberteaux, "Cross- Channel Interference Test Results: A report from the VSCA project", IEEE 802.11 Task Group p report, July 2007

[71] Available: www.its.dot.gov/aeris

[72] Ian Ku, You Lu, Mario Gerla, Francesco Ongaro, Rafael L. Gomes, Eduardo Cerqueira, Towards Software-Defined VANET: Architecture and Services, 2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)

[73] P. Pawelczak, R. Venkatesha Prasad, L. Xia, and I. Niemegeers. "Cognitive radio emergency networks-requirements and design." In the First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN), pages 601{606. IEEE, 2005.

[74] I. Akyildiz, W. Lee, and K. Chowdhury. "Crahns: Cognitive radio ad hoc networks." Ad Hoc Networks, 7(5):810{836, 2009}.

[75] D. Kreutz, F. M. Ramos, and P. Verissimo, ''Towards secure and dependable software defined networks,'' in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 55–60.

[76] R. Kloti, ''OpenFlow: A security analysis,'' M.S. thesis, Dept. Inf. Tech. Elec. Eng., Swiss Fed. Inst. Technol. Zurich (ETH), Zurich, Switzerland, 2013.

[77] S. Shin and G. Gu, ''Attacking software-defined networks: A first feasibility study,'' in Proc. 2nd Workshop Hot Topics Softw. Defined Netw., 2013, pp. 1–2

[78] K. Benton, L. J. Camp, and C. Small, ''OpenFlow vulnerability assessment,'' in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 151–152.

[79] Seyed Mohammad Mousavi, "Early Detection of DDoS Attacks in Software Defined Networks Controller", Carleton University Ottawa, Ontario, 2014

[80] Moustafa,H., Zhang,Y.: Vehicular networks: Techniques, Standards, and Applications. CRC Press, (2009).

[81] M Raya, P Papadimitratos, JP Hubaux, "Securing Vehicular Communications", IEEE Wireless Communications, Vol13, October 2006.

[82] B. Parno and A. Perrig, “Challenges in Securing VehicularNetworks”, Proc. of HotNets-IV, 2005.
[83] Parul Tyagi, Dr. Deepak Dembla, Investigating the Security Threats in Vehicular ad hoc Networks (VANETs): Towards Security Engineering for Safer on-road Transportation, 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)
[84] J. Hubaux, S. Hapkun and J. Luo, “The Security and Privacy of Smart Vehicles,” Magazine of IEEE Security and Privacy, June 2004.
[85] Available: https://www.incapsula.com/ddos/ddos-attacks/denial-of-service.html

[86]Available: http://searchsecurity.techtarget.com/definition/SYN-flooding

[87] SilviaFichera, LauraGalluccio, SalvatoreC.Grancagnolo, GiacomoMorabito, SergioPalazzo, ”OPERETTA: An OPEnflow-based REmedy to mitigate TCP SYNFLOOD Attacks against web servers”, University of Catania Dipartimento di Ingegneria Elettrica Elettronicae Informatica, v.leA.Doria6, Catania 95125, Italy, 2015

[88] T.M. Gil, M. Poletto, MULTOPS: a data-structure for bandwidth attack detection, 2001

[89] Available: https://www.nbs-system.co.uk/blog-2/ddos-how-does-it-work.html

[90] Nazrul Hoque, Dhruba K Bhattacharyya and Jugal K Kalita, Botnet in DDoS Attacks: Trends and Challenges, July, 2015

[91]Available: https://www.incapsula.com/ddos/ddos-attacks/botnet-ddos.html

[92] Q. Yan; F. R. Yu, “Distributed denial of service attacks in software-defined networking with cloud computing,” IEEE Communications Magazine, 2015, Volume: 53, Issue: 4, Pages: 52 - 59,

[93] W. Xia et al., “A Survey on Software-Defined Networking,” IEEE Commun. Surveys & Tutorials, 2014

[94] S. Scott-Hayward, G. O’Callaghan, and S. Sezer, “SDN Security: A Survey,” Proc. IEEE SDN for Future Networks and Services (SDN4FNS), 2013

[95] S. Sezer et al., “Are We Ready for SDN? Implementation Challenges for Software-Defined Networks,” IEEE Commun. Mag., vol. 51, no. 7, 2013.

[96] H. T. N. Tri and K. Kim, “Resource Attack Based On Flow Table Limitation in SDN,” no. 1, pp. 215–217

[97] C. Ji M. Thottan, "Anomaly Detection in IP Networks," IEEE Transaction on Signal Processing, vol. 51, no. 8, pp. 2291-2204, Aug 2003.

[98] D. Schnackenberg, R. Balupari, D, Kindred L. Feinstein, "Statistical Approaches to DDoS Attack Detection and Response," in DARPA Information Survivability Conference and Expedition, vol. 2003, Apr.

[99] Z. Qin, L. Ou, J. Liu, A. X. Liu J. Zhang, "An Advanced Entropy-Based DDoS Detection Scheme," in International Conference on Information, Networking and Automation, 2010, pp. 67 71.

[100] I. Ra G. No, "An efficient and reliable DDoS attack detection using fast entropy computation method," in International Symposium on Communication and Information technology, 2009, pp. 1223-1228

[101] Y. Chen X. Ma, "DDoS Detection Method Based on Chaos Analysis of Network Traffic Entropy," IEEE Communications Letters, vol. PP, no. 99, pp. 1-4, 2013.

[102] T. Nakashima, T. Sueyoshi S. Oshima, "Early DoS/DDoS Detection Method using Short-term Statistics," in International Conference on Complex, Intelligent and Software Intensive Systems, 2010, pp. 168-173.

[103] E. Mota, A. Passito R. Braga, "Lightwight DDoS flooding attack detection using 60NOX/Openflow," in IEEE 35th conference on Local Computer Networks, 2010, pp. 408-415.

[104] G. Gu S. Shin, "CloudWatcher: Network Security Monitoring Using OpenFlowin Dynamic Cloud Networks (or: How to provide security monitoring as a service in clouds?)," in 20th IEEE International conference on Network Protocols, 2012, pp. 1-6.],

[105] Jisa David, Ciza Thomas, “DDoS Attack Detection using Fast Entropy Approach on Flow-Based Network Traffic”, 2nd International Symposium on Big Data and Cloud Computing (ISBCC’15), 2015

[106] Mohamed Nidhal MEJRI, Jalel BEN-OTHMAN, “Entropy as a New Metric for Denial of Service Attack Detection in Vehicular Ad-hoc Network/ data plane communication”, C.2.0 [COMPUTER-COMMUNICATION NETWORKS]: General, Security and Protection; C.2.1

[COMPUTER-COMMUNICATION NETWORKS]: Network Architecture and Design, Network Communications

[107] M. McCauley, Available: http://www.noxrepo.org/, November, 2013

[108] Available: https://www.sdxcentral.com/resources/sdn/sdn-controllers/open-source-sdn-controllers/what-is-floodlight-controller/

[109] Available: https://www.sdxcentral.com/sdn/resources/sdn-controllers/opendaylight-controller/

[110] Available: http://mininet.org/

[111] Available: http://www.secdev.org/projects/scapy/

[112] Available: https://docs.python.org/2/library/random.html

[113] Available: http://openvswitch.org/

[114] Vishal Kumar, Shailendra Mishra, Narottam Chand, "Applications of VANETs: Present & Future", Communications and Network, 2013, 5, 12-15