SOFTWARE DEFINED SECURE AD HOC WIRELESS NETWORKS

A dissertation submitted in partial fullfillment
of the requirements for the degreee of
Doctor of Philosophy

By

MAHA ABDULLATEIF ALQALLAF
M.S., Wright State University, 20014
M.S., Ahlia University, 2009
B.S., Kuwait University, 1998

2016
Wright State University

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

April 25, 2016

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Maha Abdullateif Alqallaf ENTITLED Software Defined Secure Ad Hoc Wireless Networks BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Bin Wang, Ph.D.
Dissertation Director

Michael L. Raymer, Ph.D.
Director, Computer Science and Engineering
Ph.D. Program

Robert E. W. Fyffe, Ph.D.
Vice President for Research and
Dean of the Graduate School

Committee on Final Examination:

Bin Wang, Ph.D.

Yong Pei, Ph.D.

Krishnaprasad Thirunarayan, Ph.D.

Zhiqiang Wu , Ph.D.

ABSTRACT

Alqallaf, Maha, Ph.D., Department of Computer Science and Engineering, Wright State University, 2016. Software Defined Secure Ad Hoc Wireless Networks.

Software defined networking (SDN), a new networking paradigm that separates the network data plane from the control plane, has been considered as a flexible, layered, modular, and efficient approach to managing and controlling networks ranging from wired, infrastructure-based wireless (e.g., cellular wireless networks, WiFi, wireless mesh networks), to infrastructure-less wireless networks (e.g. mobile ad-hoc networks, vehicular ad-hoc networks) as well as to offering new types of services and to evolving the Internet architecture. Most work has focused on the SDN application in traditional and wired and/or infrastructure based networks.

Wireless networks have become increasingly more heterogeneous. Secure and collaborative operation of mobile wireless ad-hoc networks poses significant challenges due to the decentralized nature of mobile ad hoc wireless networks, mobility of nodes, and resource constraints. Recent developments in software defined networking shed new light on how to control and manage an ad hoc wireless network. Given the wide deployment and availability of heterogeneous wireless technologies, the control and management of ad

hoc wireless networks with the new software defined networking paradigm is offered more flexibility and opportunities to deal with trust and security issues and to enable new features and services.

This dissertation focuses on the SDN MANET architecture design issues for providing secure collaborative operation. Specifically, (I) We have proposed four design options for software defined secure collaborative ad hoc wireless network architecture. The design options are organized into (a) centralized SDN controller architecture with controller replication and (b) distributed SDN controller architecture. While these proposed architecture options exhibit different characteristics, many common challenges are shared amongst these options. Challenges include fault-tolerance, scalability, efficiency, and security. The unstructured nature of ad hoc wireless networks exacerbates these challenges. We have studied the pros and cons of these different design options and their applicability in different practical scenarios via simulations. (II) Establishing the initial trust among participating devices in an SDN based wireless mobile ad hoc network will serve as a basis for enabling ensuing secure communication of the network. We proposed and studied trusted virtual certificate authorities (VCAs) based local infrastructure for supporting device mutual authentication to support secure communications/operations in SDN based MANETs, and therefore, relieving the MANETs of the need to rely on an external public key infrastructure (PKI). We examined the ways in which this VCA based infrastructure can be integrated with the four SDN based MANET architecture design options. (III) Finally, we provided theoretically analysis of designing and incorporating an IDS/IPS system in an SDN based MANET.

Contents

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would never have been able to finish my dissertation without the guidance of my adviser and committee members, support from my husband, family and friends.

I would like to express my deep appreciation and gratitude to my adviser Professor Bin Wang for the patient guidance and mentorship he provided to me. Dr. Wang's patience and support helped me overcome many difficult situations and finish this dissertation. I am truly fortunate to have had the opportunity to work with him.

I would also like to thank the other members of my dissertation committee - Dr. Yong Pei, Dr. Krishnaprasad Thirunarayan and Dr. Zhiqiang Wu for their precious time in reviewing the manuscript and their valuable suggestions.

Extreme gratitude to my husband, Mohammad, I'd be remiss if I did not acknowledge his innumerable sacrifices in shouldering far more than his fair share of the parenting and household burdens while I pursued this final degree. Most importantly, none of this would have been possible without the love and patience of my family and husband.

I'm also grateful to Computer Science and Engineering department, University Center for International Education and many friends who have helped me through these four years. Their support and care helped me overcome setbacks and stay focused on my graduate

study. I greatly value their support and I deeply appreciate their belief in me.

Finally, I appreciate the financial support from the Kuwait Foundation for the Advancement of Sciences that funded the research discussed in this dissertation.

Chapter 1

Introduction

An ad hoc network is a collection of communication devices (or nodes) that can wirelessly communicate with each other without an available fixed infrastructure. These communication devices may also be mobile. In general, this type of network is known as a mobile ad hoc wireless network or MANET. Nodes in a MANET self-organize into a dynamic network in which individual nodes can transmit, receive, and/or relay information. A node in a MANET is responsible for dynamically discovering which other nodes it can directly communicate with and determining a path to reach other nodes that may be multiple hops away. Ad hoc wireless networks are suited for use in situations where an infrastructure is either not available, not trusted, costly to set up, or should not be relied on. Examples of MANETs include military ad hoc networks for soldiers in the battlefield, wireless sensor networks for environmental monitoring, networks for emergency or first responders, wireless mesh networks of rooftop-mounted ad hoc routers, and so on.

The history of mobile ad hoc wireless networks can be traced back to 1972 and the DoD-sponsored Packet Radio Network (PRNET) [74]. The research on ad hoc wireless networks has been on-going for several decades. The decentralized nature of ad hoc networks, mobility of nodes, and resource constraints have created a lot of challenges. Main-

taining security in mobile ad-hoc networks is also significantly more difficult than in wired networks due to many system constraints, such as for some mobile devices, limited power, limited wireless communication bandwidth, memory, computational power, as well as node mobility. These constraints may cause frequent network topology changes. In addition, nodes in MANETs may be subject to physical attacks, capture, and/or device compromise.

Many challenges exist in order to enable participants of an ad hoc wireless network to establish trust and for them to work collaboratively while fending off security weaknesses and attacks. Traditionally, a lack of a centralized control of an ad hoc wireless network exacerbates the problem. In following sections, we summarize the challenges that MANETs face in terms of security and trust for collaborative operations.

1.1 Security Challenges in Ad Hoc Wireless Networks

Security has increasingly been a big concern in both wired and wireless networks. In wired networks, adversaries have to go through several lines of defense to cause damages, for example, firewalls, intrusion detection and/or intrusion prevention systems. In a mobile ad-hoc wireless network, an adversary can launch attacks from anywhere in the network as long as it is within the radio transmission range of a wireless node. A lack of adequate security defense and physical security makes a mobile ad hoc wireless network an easy target of attack.

Security threats can also arise from compromised nodes inside the MANET when malicious users gain control of the nodes and use the nodes to carry out attacks. In a large MANET, numerous nodes with varying characteristics are present and it is difficult to identify where specifically the attacks originate as ad-hoc networks have no centralized

management, e.g., a server that monitors network traffic. In addition, a lack of centralized management machinery impedes trust management for nodes in a MANET. While nodes in wired networks get power from electric power supply, many nodes in ad-hoc wireless networks rely on power supplies from batteries. The restricted power supply presents opportunities for denial-of-service attacks. In addition, constrained resources such power may discourage a node in a mobile ad-hoc wireless network from cooperating with other nodes in the network. Furthermore, some nodes may selfishly choose to conserve the battery power which may jeopardize the overall security of the network. It is also difficult to maintain a network-wide infrastructure and services in MANETs, for example, a public key infrastructure, for facilitating the implementation of security mechanisms. Scalability is yet another problem that any practical security mechanism needs to address as a mobile ad-hoc wireless network scales up.

When evaluating the soundness of security of a mobile ad hoc wireless network, the general aspects/criteria of network security still apply, such as confidentiality, integrity, authentication, authorization, non-repudiation, access and availability, anonymity, and so on. Confidentiality ensures that information can only be accessible by those authorized to access. Integrity guarantees that the message transmitted have not been maliciously or accidentally altered. Authentication is assurance that the parties involved in communication are not impersonators. Authorization is a process through which entities are given access rights and privileges based their credentials. Nonrepudiation ensures that both the sender and the receiver cannot deny the actions of sending or receiving a message. Access and availability mean that a node should be always able to provide the services regardless of the security state. Anonymity ensures that all the information that can be used to identify

the owner or the user of the information is kept private.

Depending on the requirements of different applications or systems, some or many of the aforementioned security criteria need to be met to ensure the security and collaborative operations of a mobile ad hoc wireless network. To this end, it is imperative to form several lines of defense to provide an adequate level of security and protection in MANETs to deal with various types of security threats and attacks as we face increased levels of advanced persistent threats (APTs). These lines of defense range from component security mechanisms to security systems and services that can prevent, analyze, correlate, detect, and mitigate attacks.

In what follows, we outline numerous types of attacks that may occur in mobile ad hoc wireless networks. The type of attacks to which mobile ad-hoc wireless networks are vulnerable can be classified as passive or active attacks. In passive attacks, the adversary is restricted to monitoring the network communication. The purpose is to gain information about the victims without changing it. The other type of attacks is active attacks in which the adversary maliciously attempts to change or drop victims' data or inject data. An example of passive attack is wiretap attack. We can deal with passive attacks using cryptography (i.e., encrypt data to preserve confidentiality and integrity). However, cryptography alone cannot prevent some active attacks. Some of these attacks that plague mobile ad hoc wireless networks are man-in-the-middle attack, relay attack, insider attack, DoS attack, Sybil attack and rushing attack.

1.1.1 Passive Attacks

Wiretap Attack In wiretap attack the adversary intercepts the communication and tries to obtain confidential information such as public key, private key or passwords without altering it [16]. This type of attacks hurts confidentiality. The solution to this type of attacks is to use cryptography for information protection.

1.1.2 Active Attacks

Man-in-the-Middle Attack In this type of attacks, the adversary gains control to monitor and alter the information exchanged between a sender and a receiver [10].

Relay Attack A relay attack is a type of attack where the transparent adversary (man-in-the-middle) intercepts and manipulates communications between a sender and a receiver [10, 58].

Insider Attack Weaknesses in authentication mechanisms may lead to an insider attack in a MANET, where an adversary can gain access to the network and makes it look like a legitimate node [16]. The compromised node can join the network and then acts maliciously. This type of attacks may lead to other types of attacks, such as denial of service, and in the worst case may result in takeover the control of communication and/or the network.

Denial of Service (DoS) This type of attack aims at making the network service unavailable. In a wired network the attacker may flood the network with traffic to make the services unavailable [16]. A MANET is more vulnerable to DoS because of the interference-prone radio channel and the limited resources. For example, an attacker can use radio jamming and/or battery exhaustion to cause DoS in a MANET.

Sybil Attack In Sybil attack a malicious node presents one or more fake identities to other nodes in the network [26]. For example a malicious node may presents the identity of an existing node in the network to impersonate any node in the network. As another example, a malicious node may present one or more nonexistent identities and the malicious may change its identity to make its behavior difficult to trace.

Rushing Attack In rushing attack an adversary fools a sender to believe that the route is shorter by relaying packets much faster through nodes under its control. This attack may significantly influence network connectivity and weaken networking functions and capabilities such as control and message delivery [41]

1.2 Trust Management Challenges in Ad Hoc Wireless Networks

There exists a significant amount of challenge in trust management in mobile ad hoc wireless networks. Some salient characteristics of trust management in MANET environments are briefly summarized as follows [22, 30].

- Dynamicity

  Trust should be dynamic, not static. Due to node mobility or failure, network state information is typically incomplete and can change rapidly.

- Subjectivity

  MANET environments are dynamically changing. This may require a trustor node to determine a different level of trust against the same trustee node.

- Incomplete Transitivity

  Trust is not transitive. For example if node A trusts node B and node B trusts node C, this does not guarantee that node A trusts node C.

- Asymmetry

  Trust is asymmetric. For example in a heterogeneous network. The node with higher capability (e.g., higher energy capacity) may not trust nodes with lower capability at the same level that nodes with lower capability trust nodes with higher capabilities.

- Content Dependency

  Trust in MANETs may depend on the mission for which the network is tasked. Different types or levels of trust are required for different tasks.

In addition to some of the unique characteristics of trust in MANETs summarized above, any trust design for MANETs should also consider how to derive a decision procedure to determine the trust of an entity. Trust may have to be determined in a highly customizable way without causing any disruption to the devices during computation and communication. A trust decision framework should not assume that all nodes are cooperative. Some nodes may act selfishly to preserve their resources. Trust should be established in a self-organized reconfigurable manner to prevent service disruption due to the dynamics of MANETs. Moreover, trust in MANETs should consider the tradeoffs between security and performance such as scalability, reliability, fault tolerance, and resource consumption.

1.3 Countermeasures

Much research has been conducted to provide solutions to security and trust management issues in mobile ad hoc networks. Research has resulted in multiple lines of defense against both known and unknown security threats in MANETs, such as light weight authentication schemes for MANETs [93], cryptographic, integrity and non-repudiation schemes for MANETs, intrusion detection techniques, and so on. In particular, a general intrusion detection framework in MANET was proposed in [53] in which the intrusion detection module is set in each layer on each node of the mobile ad hoc network. A multi-layered integrated intrusion detection and responsive system can enhance the intrusion detection performance and countermeasures. In this architecture each node in the network is equipped with an IDS agent and all of the IDS agents can work independently and locally as well as cooperate with each other to detect intrusion behaviors in a large range [53]. Another approach is a cluster-based intrusion detection technique [42]. A MANET can be organized into a number of clusters so that every node belongs to at least one cluster and each cluster has a monitoring node called a cluster head for intrusion detection.

1.4 Software Defined Networking

Recent developments in software defined networking have shed new light on how to control and manage an ad hoc wireless network. Given the availability of heterogeneous wireless technologies, the control and management of ad hoc wireless networks with the new software defined networking paradigm is offered more flexibility and opportunities to deal with trust and security issues as well as to enable new features and services in ad hoc wireless

networks. In the following sections, we briefly review the history and concepts of software defined networking.

1.4.1 Programmable Networks

The software defined networking concept and architecture evolved from decades of research on searching for evolvable network architecture. Some early work on programmable networks that laid the foundation for many of today's ideas include open signaling [17], active networking [88], Devolved Control of ATM Networks [76], 4D project [37], NETCONF [29] and Ethane [18]. Open signaling (OPENSIG) aimed at providing access to the network hardware through open, programmable network interfaces. Beginning in the mid 1990s, active networking looked into the idea of a network that would be programmable for customized services. Devolved Control of ATM Networks (DCAN) was designed to develop infrastructure to manage and aid functions of ATM switches in the care of DCAN. The 4D project proposed a design that stressed separation between the routing decision logic and the protocols governing the interaction between network elements. NETCONF allows mechanisms to install, manipulate and delete the arrangement of network devices. It was used in routers, switches and other equipment. Ethane was the immediate predecessor to OpenFlow. It represented a new architecture for enterprise network control and management. Ethane's main focus was centered on using a centralized controller to organize policies and security mechanisms in an enterprise network [68].

1.4.2 Software-Defined Networking Architecture

The fundamental principle of software-defined networking (SDN) is to separate the control plane from the data-forwarding plane, and to allow the control of the data-forwarding

of switches/routers from a logically centralized controller using a secure communication protocol, such as OpenFlow, to communicate between the two planes.

A three-layer model for SDN architecture is shown in Figure 1.1. This architecture [33] is composed of an application plane, a control plane, and a data plane.

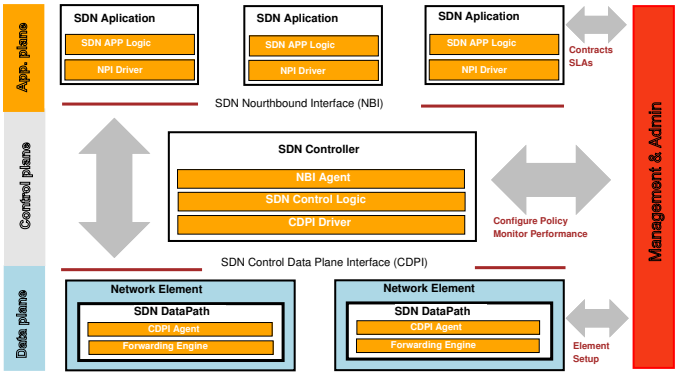

Figure 1.1: A Three-layer Model for Software Defined Networking Architecture.

- Application Plane:

  The SDN application plane is responsible for communicating the requirements of the network and the behavior to the SDN controller through the SDN North Bound Interface (NBI) drivers.

  An SDN application is made up of one SDN application logic, and one or more NBI drivers. An SDN application may reveal an additional layer of abstracted network control, thereby permitting higher-level NBIs from NBI agents.

- Data Plane:

  The data plane or data path is made up of a Control Data Plane Interface (CDPI) agent and a set of engines that forward traffic between the data path's external or internal traffic processing functions. There may be one or more data paths.

- Controller Plane:

  The SDN controller is centralized and is responsible for (1) translating the commands from the SDN application plane down to the SDN data paths, and (2) making available to the SDN applications an abstracted view of the network. An SDN controller consists of an NBI agent or agents, the SDN control logic, and the CDPI driver.

1.5 OpenFlow

1.5.1 OpenFlow Architecture

The SDN architecture discussed above in Figure 1.1 assumes the use of the OpenFlow protocol (Figure 1.2). Through OpenFlow, a controller can dictate to a network switch/router where to and how to process and/or route packets. The data path is located on the switch/router, and the switch/router and controller use the OpenFlow protocol to communicate securely with each other. An OpenFlow-based SDN network may have several physical network forwarding devices like routers, switches, virtual switches, wireless access points, and so on.

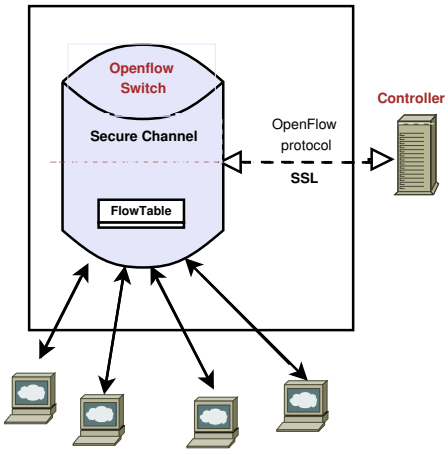

Figure 1.2: An OpenFlow Switch.

An OpenFlow switch consists of three parts as shown in Figure 1.2:

- A pipeline of flow tables with an action associated with each flow entry;
- A secure channel that connects the switch with the controller;
- The OpenFlow protocol is a standard way for the controller to communicate with the switches.

1.5.2 Flow Table

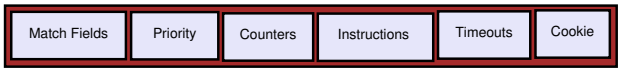

Figure 1.3: An SDN Flow Table.

An SDN switch device contains a pipeline of flow tables that consist of flow entries. As shown in Figure 1.3, each flow entry may contain (among other fields):

- Match fields: are used to match against incoming packets;
- Priority: is used to determine the precedence of a flow entry;
- Counters: when packets are matched counters are updated;
- Instructions: dictate how to handle matching packets and modify the action set;
- Timeouts: is the idle time before the flow expires;
- Cookie: is used by the controller to filter flow statistics.

The implementation of flow tables may be different from one vendor to another. Match fields and priority taken together identify a unique flow entry in the flow table. OpenFlow

provides an open protocol to program different switches and routers, enabling a network administrator to easily control the network for better performance while lowering the cost of operation and management.

1.6 SDN Applications

SDN applications were initially in enterprises like universities that run large networks. SDN has also been used in data centers like Google's internal network. SDN and OpenFlow can facilitate ubiquitous connectivity needed for cellular and WiFi wireless access networks. The work of [60] described how OpenFlow enabled networks were used in network management and access control, VLANs, mobile wireless VoIP client support, and non-IP network packet-level processing.

In network management and access control, SDN is used to allow network managers/-operators to define network policies and/or network behaviors in the central controller by making control decisions for each new flow [18]. A controller checks a new flow against a set of rules. In the SDN VLANs example, SDN can provide each user an isolated network. In the mobile wireless VoIP client example, when VoIP clients establish a new connection over the OpenFlow-enabled network a controller tracks the location of clients and re-routes connections as users move through the network, allowing seamless handoff from one access point to another. In a non-IP network application, OpenFlow-enabled switches can support non-IP traffic. For example, flows could be identified using their Ethernet header (MAC source and destination addresses). Processing packets rather than flows require every packet to be processed, e.g., an intrusion detection system that inspects every packet, or when modifying the contents of packets, such as when converting packets from one pro-

tocol format to another. Processing packets rather than flows can be done in two ways. The first approach is to force all of a flow's packets to pass through a controller by allowing the switch to forward every packet to the controller. The advantage is flexibility at the cost of performance and this provides a useful way to test the functionality of a new protocol. The second way is to route packets to a programmable switch. The advantage is that packets can be processed at line-rate in a user-definable way [60].

1.7 Security Issues in Software Defined Networks

SDN delivers agility, flexibility, new services, allowing network functions to run on off-the-shelf hardware and increased programmability of network elements to make the network simpler to design, deploy, manage and scale. However, security imposes a big challenge in software defined networks in general and in software defined ad hoc wireless networks in particular.

Two properties in software defined networking attract malicious attacks that threaten the network. The first one is the ability to control the network by means of software (which is always subject to bugs and scores of other vulnerabilities). The second is the "centralization" of the network in the controller(s). Anyone with access to the servers that host the control software can potentially control the entire network [48]. Threats and weaknesses in software defined networks are discussed below.

- Forged or Faked Traffic Flows

  This type of attacks is used by malicious or non-malicious adversaries to cause DoS attacks. The attackers can use network elements such as switches, servers, or per-

sonal devices to lunch DoS attacks against OpenFlow switches. This type of threats is not specific to SDNs, but can be a stepping stone for augmented DoS attacks.

- Attacks on Vulnerabilities in Switches

  This type of attacks appear as abnormal behaviors of one or more network devices, for example one switch may drop, slow down packets, or inject traffic or forged packets to overload the controller. This threat is not specific to SDNs. However, the impact on SDN is potentially more severe.

- Attacks on Control Plane Communications

  Man-in-the-middle vulnerable implementations of TLS/SSL can be exploited to attack the communications between controllers and switches. This threat is specific to SDN where communication with logically centralized controllers can be explored.

- Attacks on Vulnerabilities in Controllers

  This type of attacks is the most dangerous to SDN because a faulty or malicious controller could compromise an entire network. This threat is specific to SDNs as controlling the controller may compromise the entire network.

- Lack of Mechanisms to Ensure Trust between the Controller and Management Applications

  This threat is specific to SDNs. Malicious applications can now be easily developed and deployed on controllers due to a lack of mechanisms to ensure trust between the controller and management.

- Attacks on Vulnerabilities in Administrative Stations

  This threat is common in traditional networks and is exploited in SDNs to access the network controller. The difference is that the impact of this threat increases dramatically in SDNs as in SDN it becomes easy to reprogram the network from a single location.

- Lack of Trusted Resources for Forensics and Remediation

  Reliable information from all components and domains of the network is important for investigating a security incident. Trusted data and resources are required to allow efficient recovery of network elements. This threat is not specific to SDNs, but it is critical to ensure the diagnosis of and the fast recovery from network failures.

As discussed above, some of the threats and weaknesses also present in traditional networks. However, they are augmented in the case of SDN. Other threats are specific to SDN as they arise from the separation of the control and data planes as well as the presence of the logically centralized controller.

1.7.1 Mechanisms for Security Enhancement

With all the potential security issues discussed above, we enumerate some general solutions to address the threats and weaknesses identified in SDNs, such as replication, diversity, trust between devices and controllers, trust between applications and controllers software, security domains, fast and reliable software update and patching [48]. Nevertheless, employing traditional techniques, such as firewalls or intrusion detection systems, packet-forwarding policies [32], will also benefit SDN based network design.

1. Replication

   Replicated controllers instances can ensure fault tolerance in hardware and software. Replication makes it possible to mask failures and to isolate malicious or faulty applications and/or controllers, and scale up the network performance. Moreover in case of a network partition, there are controllers in different partitions.

2. Diversity

   Replication with diverse controllers can improve the robustness of secure and dependable systems. The basic principle behind this mechanism is to avoid common-mode faults (e.g., software bugs or vulnerabilities). For example, OS diversity constrains the overall effect of attacks on common vulnerabilities.

3. Trust between Devices and Controllers

   An important requirement for a control plane is to establish trust with other devices in the network. A simple approach is to have a list of authenticated trusted devices tracked by the controller. Another approach is to trust all devices until abnormal behavior is reported.

4. Trust between Applications and Controller Software

   Changing behaviors of software components due to aging, exhaustion, bugs, or attacks require trust mechanisms between applications and controller's software. In [96] the authors propose a dynamic trust model in which a trustor measures a trustee's behavior based on quality attributes, such as availability, reliability, integrity, safety, maintainability, and confidentiality.

5. Security Domains

   The use of isolated security domains is a common technique used in different types of security systems [48]. For example, sandboxing and virtualization can be used in SDN control platforms to enable the design of strong isolation modes.

6. Secure Components

   Secure and dependable system security components are one of the essential building blocks. They provide trusted computing bases and assure security properties such as confidentiality. An example is a tamper-proof device used to store sensitive security data. In case when the system is compromised, sensitive data will still be secured.

7. Fast and Reliable Software Update and Patching

   A control platform should be deployed with mechanisms to assure a smooth and secure way of receiving updates and patches [71].

1.8 Related Work to the Dissertation

Software defined networking can bring many benefits to wireless and mobile networks such as easier deployment of new services, reduced management and operational costs of heterogeneous technologies, efficient operation of multi-vendor infrastructure, increased accountability and service differentiation, continuous and transparent enhancement of network operation. Because of these potential advantages, the research community is paying increasing attention to SDN.

1.8.1 SDN Mobile Cloud

Software-defined networking provides management constructs that add capabilities for data centers both local and connected via wide-area networks. The result is a dynamic, flexible cloud infrastructure. There are challenges in applying software-defined networking in mobile cloud on top of a MANET. These challenges include (1) the wireless medium creates unreliability; (2) The mobility of nodes creates complexity. Both challenges need mechanisms that are not required in wired networks. In [49] the authors propose a framework for SDN-based mobile cloud that includes the components required by SDN in an ad hoc environment. Their architecture is based on frequency selection that allows for many types of new services using a wireless SDN-based mobile cloud. They compare their system with traditional MANET routing to determine how SDN can help overcome unreliable wireless links and mobility in the SDN-based mobile cloud.

1.8.2 Sensor OpenFlow

Wireless sensor networks in general face three problems: (1) application-specific, (2) rigidity to policy changes, and (3) difficulty to manage. These problems are inherited to wireless sensor networks. Each node is fully fledged with all physical up to application layer functionalities. In general the current wireless sensor network architecture lacks good abstraction and carries too much complexity, making WSN unwieldy, inelastic to changes and hard to manage. In [57] the concept of software defined wireless sensor network and sensor OpenFlow were proposed. This research was the first effort that provide synergies between software-defined networking and wireless sensor networks. The work aimed at developing wireless sensor networks that are versatile, flexible, and easy to manage.

1.8.3 Software Defined Wireless Networks

[39] proposed a routing protocol that considers node energy, load balancing, and network topology changes in an SDN-based wireless sensor network. Unbridling SDNs [24] is a first attempt that aimed at analyzing how SDN can be beneficial in wireless infrastructure-less networking environments with special emphasis on wireless personal area networking. The key idea of MobileFlow [70] consists of the MobileFlow forwarding engine (MFFE) and MobileFlow controller (MFC). MFFE features a lightweight protocol to allow MFFE to communicate with an MFC. The main functionalities of the MobileFlow controller (MFC) include topology auto-discovery, topological resource view, network resource monitoring and network resource virtualization. Moreover the MFC is responsible for network functions such as tunnel processing, mobility anchoring and routing. The proposed software defined networking architecture, MobileFlow, aims at future carrier networks by building on the decoupling of data and control in the mobile network user plane and a new MobileFlow stratum, which can significantly increase the operator innovation potential.

1.9 Organization of Dissertation

Wireless networks have become increasingly more heterogeneous [68]. SDN has been considered as an efficient paradigm to aid the deployment and management of network applications and services to interconnect users and applications over networks ranging from wired, infrastructure-based wireless (e.g., cellular wireless networks, WiFi, wireless mesh networks), to infrastructure-less wireless networks (e.g. mobile ad-hoc networks, vehicular ad-hoc networks). However, most current work focused on SDN in infrastructure networks.

The absence of SDN in infrastructure-less network such as mobile ad-hoc networks, vehicular networks is because a centralized control mechanism may be ill-suited to the level of decentralization, disruption, and delay present in infrastructure-less environments [68].

In this work, we intend to apply SDN to enable more collaborative and secure mobile ad hoc wireless networks while taking advantage of heterogeneous wireless networking technologies for enabling SDN-based control of such networks.

SDN for infrastructure-less networks such as mobile ad-hoc networks or heterogeneous networked environments needs to address challenges of infrastructure based SDN and the challenges for independent forwarding devices and controllers in infrastructure-less networks. The challenges for infrastructure based SDN include deploying a framework on end devices, handling multiple domains of control, supporting a flexible set of rules and actions, recognizing diverse device capabilities, tolerating delay and disruption, and integrating with other control planes. The challenges for infrastructure-less based SDN are security, trust, compatibility, maintaining the balance of control and flexibility.

The rest of the dissertation is organized as follows. Chapter 2 describes the software defined wireless network architecture design options. Chapter 3 depicts the trust management in wireless ad hoc network and discusses trust-enabling mechanisms based on virtual authentication authority. Chapter 4 presents intrusion detection and prevention systems (IDS/IPS) for software defined mobile ad hoc wireless networks. Chapter 5 presents design options customization and implementation. In Chapter 6 we report simulation based evaluation experiments, results and discussions. Chapter 7 concludes the dissertation with possible future work.

Chapter 2

Software Defined Wireless Network Architecture

2.1 Introduction

Due to the decentralized nature of mobile ad hoc wireless networks, mobility of nodes, and resource constraints, maintaining security in mobile ad hoc wireless networks poses significant challenges and may result in low network performance and high overhead. Yet, security provisioning in mobile ad hoc wireless networks is an important issue for most practical applications.

Recent developments in software defined networking have shed new light on how to control and manage an ad hoc wireless network. However, not much research has been conducted in this area. Many unanswered questions exist. For example, how should SDN concepts be adapted to suit the characteristics of unstructured wireless networks such as mobile ad hoc wireless networks?

The focal point of this dissertation is to propose, study, and evaluate design choices for software defined mobile ad hoc wireless networks to provide better support for security and trust management. Specifically, in this chapter, we will present four design options

of SDN architecture for collaborative secure ad hoc wireless networks. These options are categorized in two types: (1) centralized SDN controller with controller replication; and (2) distributed SDN controller architecture.

In centralized SDN controller with controller replication architecture, the SDN controller is effectively the brain of the network. The controller determines the collaborative behavior of the network and instructs the participating nodes how to behave. The controller runs the network control functions and the control functions create the states on the nodes. However with a centralized controller come challenges. Challenges include fault-tolerance, scalability, efficiency, and security. The unstructured nature of ad hoc wireless networks exacerbates these challenges. We will propose and study two design options to deal with these challenges (D1) fully centralized SDN controller with controller replication without infrastructure; and (D2) fully centralized SDN controller with controller replication operating in heterogeneous networking environments, such as cellular 4G and/or WiFi.

The distributed controller architecture aims at improving control plane availability and scalability. The distributed controller architecture has two different design options, (D3) hierarchical physically distributed SDN controllers to achieve a logically centralized controller; and (D4) fully distributed but logically centralized SDN controllers. In both scenarios, an SDN controller is implemented in multiple servers located in the network and collectively provides a programmatic interface (i.e., a control logic) to implement network control and management services such as routing, access control, trust, and security.

2.2 Centralized SDN Controller with Controller Replication

In this architecture, the control and management of the ad hoc wireless network is centrally located at the SDN controllers. The controller is effectively the brain of the network. The controller determines the collaborative behavior of the network and instructs the participating nodes how to behave. The controller runs the network control functions and the control functions create the states on the nodes. We propose and describe two architecture design options for this centralized SDN controller architecture: (D1) fully centralized SDN controller with controller replication without infrastructure; and (D2) fully centralized SDN controller with controller replication operating in heterogeneous networking environments, such as cellular 4G and/or WiFi.

2.2.1 Centralized SDN Controller with Controller Replication without Infrastructure

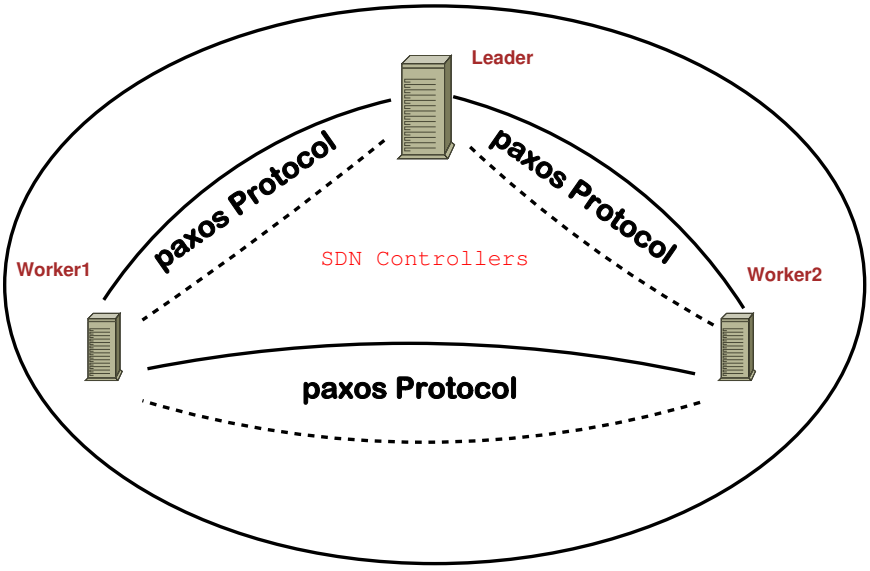

Figure 2.1: Centralized SDN Controller Architecture without Infrastructure.

Figure 2.1 shows the overview of the architecture design option (D1) of a centralized SDN controller with controller replication architecture without supporting network infrastructure. This network architecture is applicable for independent ad hoc wireless mobile networks without external supporting network infrastructure.

Independent ad hoc wireless mobile networks without external supporting network infrastructure can be considered as the purest form of MANET. The studies on this type of network has been abundant in the open literature. The extension of SDN to this type of networks calls for the consideration of fault tolerance and operating efficiency in terms of the overhead of controlling and managing the network.

This architecture may have M controllers (e.g., $M = 3$). One of the controllers acts as the leader controller and $M - 1$ controllers as workers. All controllers maintain the same state. Participant nodes interact with the leader. The network control state changes do not happen until all M controllers are in agreement. In case that the leader fails, the remaining $M - 1$ controllers work together to elect a new leader. The synchronization of controllers can be achieved using a synchronization protocol such as an adapted Paxos [50] for wireless environments. Below we illustrate the architecture and the SDN controller design using Paxos as an example.

1. Design Overview

   The centralized SDN controller will be selected and designated to run on a few nodes in the MANET. The numerous node selection algorithms can be used and the discussion is out of the scope of this work. Depending on applications, the role of the SDN controller can be assumed in a round-robin fashion to distribute the responsi-

bility and the resource consumption among the selected nodes. The control plane communication among the SDN controllers as well as communication between the controllers and the MANET nodes can take place in-band or out-of-band, depending on the hardware provisioning of the network. The communication efficiency for both control and data planes can be enhanced by having multiple network interfaces so that the traffic of the two planes do not interfere with each other by using different channels, e.g., using different frequency bands.

The states of the SDN controller can be replicated to a few additional backup controllers. A synchronization protocol such as an adapted Paxos can be used for such purpose. Paxos "is a family of protocols for solving consensus in a network of unreliable processors. Consensus is the process of agreeing on one result among a group of participants" [50]. Paxos provides safety (consistency) for a fault-tolerant distributed consensus. Paxos specifies the actions of the participating processes by their roles in the synchronization protocol. Paxos roles include client, acceptor, proposer, learner, and leader. A single process may play one or more roles at the same time.

Paxos Roles

(a) Client

The client role is to issue a request to the distributed system, and wait for a response. For example, a client requests a write permission on a file in a distributed file server.

(b) Acceptor

"Acceptors are collected into groups called quorums. Any message sent to an acceptor must be sent to a quorum of acceptors. Any message received from an acceptor is ignored unless a copy is received from each acceptor in a quorum" [50]. Quorums ensure that at least some surviving processor retains knowledge of the results to show the safety property of Paxos.

(c) Proposer

A proposer is a coordinator between clients and acceptors, aiming at moving the protocol forward when conflicts occur. Moreover a proposer supports the client request and tries to convince the acceptors to agree on it.

(d) Learner

When the client request has been agreed upon by the acceptors, the learner sends a response to the client indicating that his request is approved. A learner could be considered as a replication factor for Paxos. In addition more learners could be added to improve availability of processing.

(e) Leader

"Paxos requires a distinguished proposer (called the leader) to make progress" [50]. Many proposers believe that they are leaders. They may try to stall the protocol by proposing conflict updates. However Paxos guarantees progress if one leader is eventually chosen. In our proposed architecture, the controller is a leader.

(f) Quorums

Quorums contain subsets of acceptors. Any two subsets must share at least one

member. For example if the set of acceptors are (X, Y, Z), a majority quorum would be any two acceptors $(X, Y), (X, Z), (Y, Z)$. A quorum is any majority of participating acceptors [50].

2. Paxos Operations

Paxos works in several rounds. A successful round has two phases:

Phase 1

(a) Prepare

A number N (the proposal identification) is created by a proposer (the leader or the controller). The number N must be greater than any previous proposal number used by this proposer. The proposer sends a prepare message containing this proposal to a quorum of acceptors. The proposer has the right to decides who is in the quorum.

(b) Promise

A promise is returned by an acceptor to ignore all future proposal that has a number less than the current number N if the proposal number N received is higher than any previous proposal number received by from any proposal by the acceptor. If the proposal number is equal to a proposal number in the past, the acceptor has the right to accept the received proposal under one condition that is to include the previous proposal number and previous value in its response to the proposer. Or the acceptor can ignore the received proposal. In this case for the purpose of optimization, sending a denial (NACK) response would tell the

proposer that it can stop its attempt to create consensus with proposal N.

Phase 2

(a) Accept Request

A proposer sets a value to its proposal when it have received enough promises from a quorum of acceptors. If any acceptors had previously accepted any proposal, then they will have sent their values to the proposer, who now must set the value of its proposal to the value associated with the highest proposal number reported by the acceptors. If none of the acceptors had accepted a proposal up to this point, then the proposer may choose any value for its proposal. The proposer sends an accept request message to a quorum of acceptors with the chosen value for its proposal.

(b) Accepted

If an acceptor receives an accept request message for a proposal N, it must accept it if and only if it has not already promised to only consider proposals having an identifier greater than N. In this case, it should register the corresponding value v and send an accepted message to the proposer and every learner. Else, it can ignore the accept request. In addition an acceptor can accept multiple proposals. These proposals may even have different values in the presence of certain failures. However, the Paxos protocol will guarantee that the acceptors will ultimately agree on a single value.

3. Paxos Fault-tolerance

Paxos faces three major faults: failure of an acceptor, failure of a redundant learner, and failure of a proposer.

(a) Failure of Acceptor

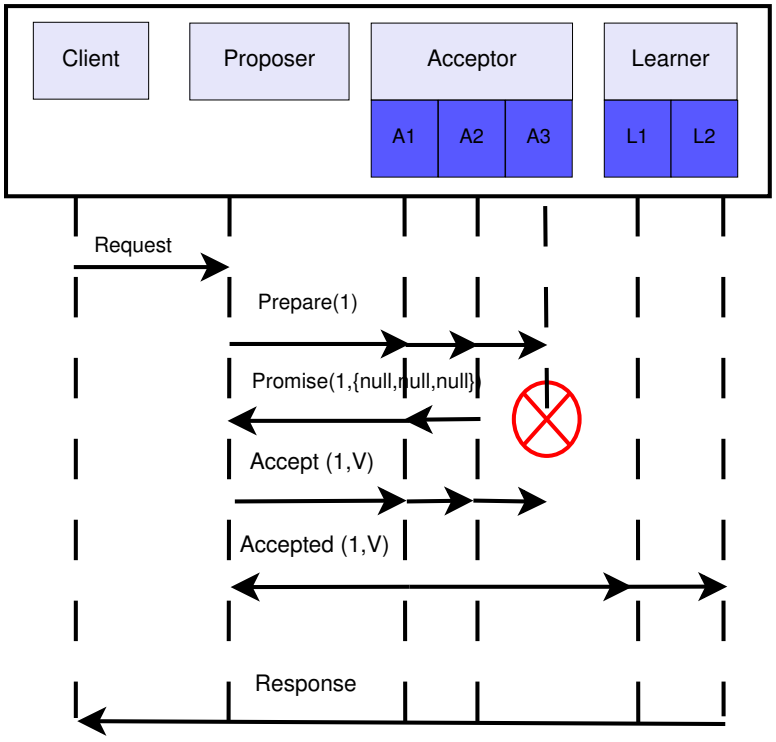

Figure 2.2: Failure of Acceptor.

When the clients send commands to a leader (proposer), the leader (proposer) sends prepare to a quorum of acceptors. If one of the acceptor failed and the leader (proposer) received promises from the majority of acceptors in the quorum, the proposer sends an accept request message to a quorum of acceptors with the chosen value for its proposal.

If a failure of an acceptor occurs when a quorum of acceptors remains live, the protocol requires no recovery. No additional rounds or messages are required, this type of failure is known as simplest error case. As shown in Figure 2.2.

(b) Failure of Redundant Learner

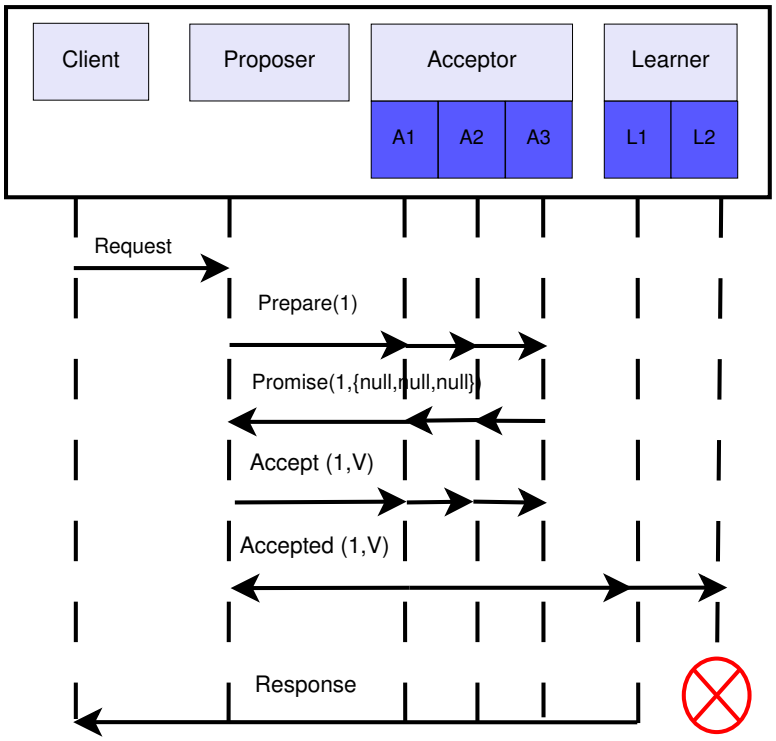

Figure 2.3: Failure of Redundant Learner.

When one learner fails, the protocol requires no recovery. No additional rounds or messages are required, this type of failure is also known as simple error case. As shown in Figure 2.3.

(c) Failure of Proposer

When a leader (Proposer) fails after proposing a value, but before agreement is reached. A second leader (proposer) is elected, Figure 2.4 shows the message flow. The problem occurs when the primary leader recovers and thinks that he is still a leader (proposer). The recovered leader has not learned that a new leader is elected yet and attempts to begin a round in conflict with the current leader. In this case, another round must be started with a higher proposal number.

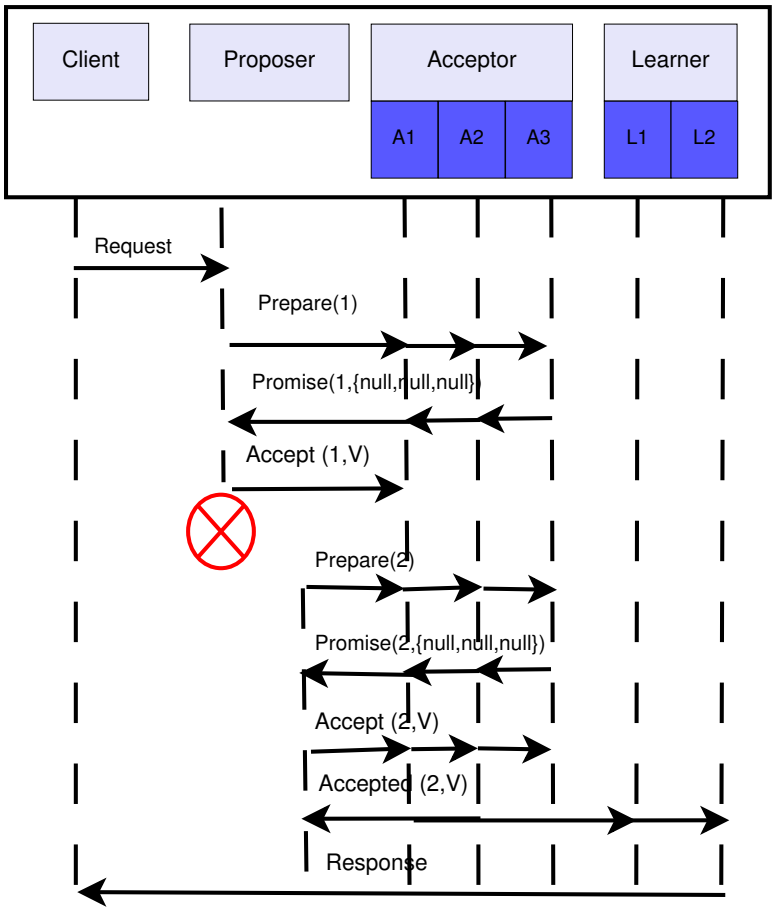

Figure 2.4: Failure of Proposer.

4. Paxos Implementation in Ad Hoc Networks

   Some past research has shown that Paxos based synchronization can be adapted to deal with state consensus in MANETs. In [12], the authors proposed the Paxos/lastvoting algorithm. Lastvoting is basically Paxos expressed in the HO model [20]. The idea behind this model is asynchronous rounds. In each round every process sends messages and receives messages. Lastvoting algorithm works in four rounds. In the first round each process sends proposal and a time stamp to the leader. If the leader receives proposals from the majority it will set its proposal number with the highest time stamp. In the second round the leader sends the vote for all. All processes update their proposal number with the time stamp when receiving the vote from the leader. In the third round all processes send (ACK) to the leader when they received majority of (ACK). In the fourth and the last round the leader sends its decision to all processes. The work in [12] shows that adequate communication layer can handle communication in multi-hop ad hoc networks without any additional overhead for the routing of messages and election of leader.

2.2.2 Centralized Controller with Controller Replication in Heterogeneous Environments

As wireless networking technologies become increasingly diverse, at any time and any place, different types of wireless networks may be available simultaneously such as WiFi, WiMax, cellular wireless networks, or other technologies. These infrastructure based ubiquitous networks may be integrated with the ad hoc wireless networks to form a heterogeneous wireless environment and multiple networking technologies can be used to comple-

ment each other, for example, traffic offloading for SDN control plane.

1. Design Overview

   This architecture (D2) is inspired by software defined network where the control plane and the data plane are decoupled in the mobile devices, and the control plane is shifted to a centralized controller out side of the MANET. The controller, reachable via the heterogeneous wireless network, manages a collection of mobile devices by informing each device how to handle traffic based on neighbor information provided by the mobile devices. The architecture consists of a central controller, mobile devices, and a suite of communication protocols among the controller and devices, and among mobile devices.

   SDN Controller The essential goal of the SDN controller is to have a network-wide view of the network and to provide efficient, cost-effective, and scalable network control and management. This is accomplished by communicating with and collecting information from each mobile node. In this architecture there are two types of communication. One is between the controller and the mobile nodes and the other type is among mobile nodes. In the mobile ad-hoc wireless network, nodes are equipped with multiple interfaces for communication in the heterogeneous environment. Interfaces that have access to the ubiquitous wireless networks such as WiFi, 4G, are used to communicate with the controller. Network state information will be collected and sent to the controller, such as the node's ID, neighbor updates, keep-alive messages, and so on. The controller uses the collected information to create a

global view of the network and determine network control and management information, such as the routes among nodes, and sends the information back to program the nodes.

Nodes Nodes are equipped with multiple interfaces. The interfaces connected to the heterogeneous network are used to send information to the controller such as the node's ID, neighbor updates, and keep-alive messages. At the same time, the same interface is used to receive information from the controller, such as IP assigned and routing tables. Other interfaces are used for local data plane communication among the nodes to send and receive periodic neighborhood discovery messages, such as Hello messages. The interfaces are also used for sending, receiving, and forwarding data packets.

Communication In this architecture a general communication layer is needed for heterogeneous wireless networks where multiple networking technologies (e.g., WiFi, 4G, MANET) are used to complement each other. There are two type of communications. One is between the controller and the mobile nodes (for SDN control messages) and the other type is among mobile nodes (for data forwarding).

Replication of Controllers Centralized network management in this architecture may simplify the task of managing the network as well as achieves better performance for both control and data forwarding. However, the vulnerability of controller failure still exists. This architecture requires to configure one or more backup con-

trollers which can assume the network control in case of a SDN controller failure.

2. Implementation of Controller in Heterogeneous Environments

   To enable such an SDN architecture in a heterogeneous wireless environment, the communication layer needs to be extended and the controller needs to deal with heterogeneous technologies. As a result, the mobile devices themselves do not have to run any routing protocols. Instead, the centralized controller handle this service, transferring the duty of network management calculations from battery constrained devices to a powerful server and over a separate communication channel. Figure 2.5 shows an overview of the controller.

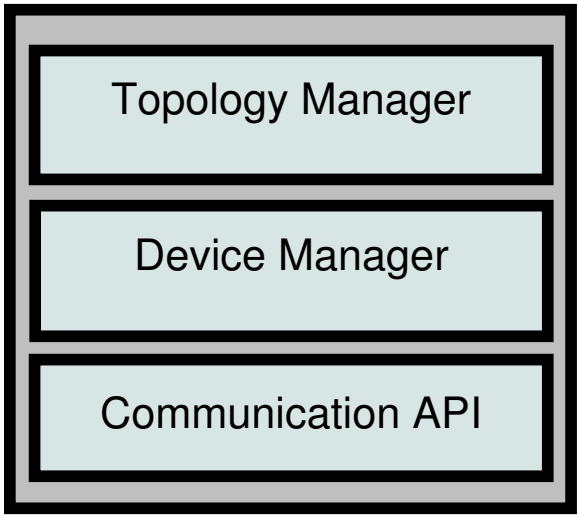

Figure 2.5: An Overview of the SDN Controller for MANET in Heterogeneous Environments.

Controller The controller contains a few components: (1) Communication API, the API enables the controller to communicate with the mobile nodes, collect information for each node such as node's ID, neighbor information, interfaces, connectivity; (2) Device Manager is responsible for managing information collected.

As a result, the controller has a dynamic view of the available mobile devices; (3) Topology Manager, through topology manager the controller obtains a network-wide view. Routing information for data communication is calculated based on information gathered. The topology manager sends the routing information back to the nodes for populating their routing tables.

Nodes Each node in the SDN network maintains two tables. The first one is the neighbor list table. The table contains neighbor's ID and ID (assigned by the controller) and its time stamp. Every time the node receives periodic Hello message from neighboring nodes. Upon receipt of a Hello message, the node updates the table by either adding the neighbor, if it does not exist already, or refreshing the neighbor's time-stamp. If a timer of one of the neighbors times out, the node deletes the neighbor from its list. In case of adding or deleting a neighbor, the node updates the controller with such information. The second table is the forwarding table. It contains next-hop (neighbor ID) and destination ID fields. The forwarding table is updated whenever the controller sends a route update to the node. Once the node receives a data packet to be forwarded, it looks at the packet's header and matches the destination ID with the corresponding next-hop in the routing table. The node then uses the neighbor list to look up the next-hop ID and forward the packet to it through the local communication channel.

Communication Mobile nodes have a communication layer and this layer provides low level functions such as sending or receiving a message to or from a node, and

sending or receiving a message to or from the SDN controller using different communication interfaces. To reduce the number of messages sent from nodes and the SDN controller, two approaches can be exploited. First, the node does not need to send keep-alive messages if it has recently sent the controller a neighbor update message. Once the controller receives any message from the node, it updates its timer and considers the node reachable. Second, after the controller calculates the routes, rather than sending the entire routing table of the network to all nodes, it only sends routing table updates to the nodes affected by the topology change.

2.3 Distributed SDN Controller Architecture

The second category of SDN based MANETs use a set of distributed SDN controllers. This type of network architecture aims at improving the network control plane availability and network scalability. For this type of distributed controller architecture, we propose and study two different design options: (1) using hierarchical physically distributed SDN controllers to achieve a logically centralized controller (D3); and (2) using fully distributed but logically centralized SDN controllers (D4). In both scenarios, the SDN control platform is implemented in multiple servers located in the network and collectively provides a programmatic interface (i.e., a logically centralized control logic) to implement network control and management services, such as routing, access control, and so on.

2.3.1 Hierarchical Distributed SDN Controllers

1. Design Overview

   This architecture defines a control hierarchy of main SDN controller (MC) and sec-

ondary SDN controllers(SCs) [78]. These SDN controllers are distributed. However, this distributed SDN control enables a logically centralized control through control delegation between different levels of the control hierarchy.

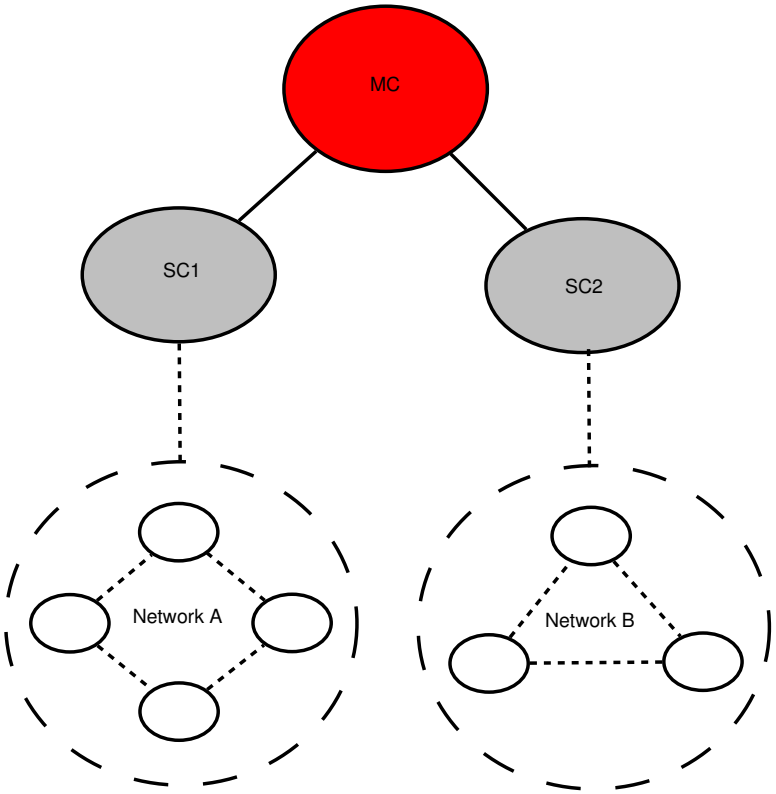

Figure 2.6: A Hierarchy of SDN Controllers.

Hierarchy of SDN Controllers A hierarchy of SDN controllers is shown in Figure 2.6 with two levels of controllers (MC and SCs). The architecture works by delegating some control functions from the MC to the SCs. SCs should authenticate with the MC before they work together.

Control Delegation and Scalability Control delegation can be accomplished in two ways. Either the MC initiates the delegation of control to the SCs or the SCs

request for delegation. Delegation requests can be triggered by different types of events. For example, the SCs in MANETs can request a control delegation from the MC when a node in its close geographical area requests to join the network. With controller delegation, the network can be organized to deal with large number of nodes while still being able to absorb the control overhead of such large-scale networks.

MC-SC Communication MC-SC communication occurs within the same administrative domain and may proceed in two steps. First, an SC requests authorization for managing a specific SDN-enabled device (i.e., Check-in Request). Second, the MC, upon accessing its database, authorizes or denies access by the requesting SC (i.e., Check-in Response).

2. Fault-tolerance

The fault-tolerance of controllers in this architecture can be implemented as follows. The MC sends periodic Hello messages as keep-alive signaling. When the SCs detect that the current MC fails after not receiving Hello messages for a predefined period of time, SCs will start an election process to select a new MC. If a new MC is elected, it will inform the corresponding devices under its control. The MC also sends Update messages to SCs to make sure that their states are in synchronization.

2.3.2 Fully Distributed SDN Controllers

1. Design Overview

In this architecture, a fully distributed control logic running in multiple servers lo-

cated in the network to collect information and provide a programmatic interface (control logic) to implement network control and management services such as routing, access control, and so on. An example of distributed SDN controller is the Onix [47], a platform on top of which a network control plane can be implemented as a distributed system running in one or more physical servers. Onix is responsible for giving the control logic programmatic access to the network.

2. Fault-tolerance and Scalability

Onix failures To provide better fault-tolerance, Onix uses more than one instance for the simultaneous management of a network element. If one of the instances fails, the failure is detected by the control logic. Similar to controller delegation, using fully distributed SDN controllers, the network can be organized to deal with large number of nodes while still being able to absorb the control overhead of such large-scale networks.

Connectivity Infrastructure Failures Source routing combined with multipath routing can be used, for example, in Onix, to reestablish connectivity when the communication between the controller and network element fails, i.e., source routing of packets over multiple paths guarantees reliable connectivity to the managed network elements as well as between Onix instances [47].

2.4 Summary

In this chapter, we have proposed four design options for software defined secure collaborative ad hoc wireless networks. The design details of the proposed SDN architecture are presented. The design options can be organized into (1) centralized SDN controller architecture with controller replication and (2) distributed SDN controller architecture. While these proposed architecture options exhibit different characteristics, many common challenges are shared amongst these options. Challenges include fault-tolerance, scalability, efficiency, and security. The unstructured nature of ad hoc wireless networks exacerbates these challenges. We will study the pros and cons of these different design options and their applicability in different practical scenarios via simulations and results will be presented in Chapters 5 and 6.

Chapter 3

Trust Establishment in Software Defined Secure Ad Hoc Wireless Networks

3.1 Introduction

Making participants in an ad hoc wireless network to securely collaborate with each other is inherently difficult to accomplish. There are many reasons. Many wireless nodes are severely resource constrained, e.g., resource constrained sensor nodes. This rules out many trust management and security schemes that require significant resource commitments. Nowadays, devices are heterogeneous, of different types, with divergent capabilities, and produced by different manufacturers, for example, medical devices in a personal area wireless network. Additionally, participant devices should be considered as independent entities due to the dynamic nature of a MANET and these devices can join and leave the network at any time.

As discussed in Chapter 1, due to the inherent weaknesses of software defined networking architecture, trust and secure communication need to be established between the SDN controller and other participant devices. In general, regardless whether SDN is applied or not, MANET participants need to establish initial trust among themselves to bootstrap trust

management, security provisioning, and to enable effective collaboration.

This chapter studies the initial trust establishment approaches. A simple approach is pre-distribution of shared keys to all participants. This approach is suitable for small and static networks, however may not be feasible to deal with network dynamics. Manual device (and key) configuration in the field is also not feasible and scalable due to the dynamic and ad hoc nature of MANETs. Another approach is to deploy and use a public key infrastructure and implement mutual authentication via PKI. This approach can be resource demanding and costly, and maintaining a PKI may over burden the network itself. A more feasible PKI based approach is to reduce the overhead of maintaining a physical PKI and instead to use a virtual public key infrastructure. This is the approach we will propose and elaborate in this chapter. Yet another approach is to use zero knowledge proof to establish initial trust among participants.

The rest of the chapter is organized as follows. In Section 3.2, we will present an overview of trust management approaches, challenges, attacks on trust management schemes in MANETs some related work, and specifically the initial trust and security establishment problem in SDN MANETs. In Section **??**, we will propose a lightweight initial trust establishment scheme based on authentication using a virtual certificate authority for SDN based MANETs.

3.2 Overview of Trust Management and Challenges

There are many different definitions of trust. In one respect, trust can be understood as "a set of relations among entities that participate in a protocol. These relations are based on the evidence generated by the previous interactions of entities within a protocol. In

general, if the interactions have been faithful to the protocol, then trust will accumulate between these entities [30]." We focus on the initial trust establishment among entities before real communication among them starts. This initial trust will serve as the foundation for providing the network-wide secure communication.

3.2.1 Trust Management

The scope of network trust management is very broad, ranging from authentication to various aspects of communications and networking such as secure routing, intrusion detection, access control and key management. Trust management consists of three parts: trust establishment, trust update and trust revocation. Trust establishment includes collection of appropriate trust evidence, trust generation, trust distribution, trust discovery, and evaluation of trust evidence. Some trust establishment frameworks are listed below:

- Reputation-based Framework

  In [77], a reputation-based framework uses direct observations and second-hand information distributed among nodes in a network to evaluate a node's trustworthiness.

- Trust Establishment-based Framework

  A trust establishment framework evaluates neighboring nodes based on direct observations while trust relations between two nodes without prior direct interactions are built through a combination of opinions from intermediate nodes [77].

- Policy-based Framework

  Policy-based trust management is based on strong and objective security schemes such as logical rules and verifiable properties encoded in signed credentials for con-

trolling users' resource access. In addition, the access decision is usually based on the mechanism that has a well defined trust management language with strong verification and proof support [31].

- Evidence-based Framework

  Evidence-based trust management considers anything that proves trust relationships among nodes. These could include public keys, addresses, identities, or any evidence that any node can generate for itself or other nodes through a challenge and response process [51].

- Monitoring-based Framework

  Monitoring based trust management rates the trust level of each node based on direct information (e.g., observing the benign or malicious behaviors of neighboring nodes, such as packet dropping, and packet flooding leading to excessive resource consumption in the network or denial of service attack), as well as indirect information (e.g., reputation ratings, such as recommendations forwarded from other nodes) [51].

- Certificate-based Framework

  Certificate based trust management is a pre-deployment knowledge of trust in the network. Trust decisions can be made based on a valid certificate that proves trustworthiness of the target node by a certificate authority or by other nodes that the issuer trusts [4].

- Behavior-based Framework

  In behavior-based framework, each node continuously monitors behaviors of its

neighboring nodes in order to evaluate trust. The behavior-based framework is a reactive approach, operating under the assumption that the identities of nodes in the network are ensured by preloaded authentication mechanisms. For example, if a node uses network resources in an unauthorized way, it will be regarded as a selfish or malicious node, and may finally be isolated from other nodes [4].

- Hierarchy-based Framework

  Hierarchy exists among the network nodes based on their capabilities or levels of trust. Centralized certificate authority provides online and offline evidence [4].

3.2.2 Attacks on Trust Frameworks in MANETs

There are numerous types of known attacks against trust mechanisms in MANETs.

Bad Mouthing Attack In bad mouthing attack, a malicious user intentionally provides bad recommendation regarding other nodes [87].

On-Off Attack In on-off attack, a malicious node behaves normally and maliciously in an alternative manner to ensure damages go undetected [87].

Conflicting Behaviour Attack In conflicting attack, an attacker can behave inconsistently. For example attacker can behave maliciously in dealing with one group of nodes and normally with another group of nodes. This will lead to the performance degradation of the trust management [28].

Camouflage Attack In camouflage attack, the malicious node provides recommendation based on the majority view and then provides false information to degrade the trust scheme.

Collusion Attack A collusion attack consists of a group of nodes collaborating to provide false information regarding a normal honest node.

3.2.3 Trust Management Challenges in MANETs

There exist a lot of challenges in trust management in mobile ad hoc wireless networks. Some salient characteristics of trust management in MANET environments are briefly summarized as follows [22, 30].

- Dynamicity

  Trust should be dynamic, not static. Due to node mobility or failure, network state information is typically incomplete and can change rapidly. Trust management must be able to adapt to these changes.

- Subjectivity

  MANET environments are dynamically changing. This may require a trustor node to determine a different level of trust towards the same trustee node.

- Incomplete Transitivity

  Trust is not transitive. For example if node A trusts node B and node B trusts node C, this does not guarantee that node A trust node C.

- Asymmetry

Trust is asymmetric. For example in a heterogeneous networking environment, the node with higher capability (e.g., higher energy capacity) may not trust nodes with lower capability at the same level that nodes with lower capability trust nodes with higher capabilities.

- Content Dependency

  Trust in MANETs may depend on the mission for which the network is tasked. Different types or levels of trust are required for different tasks.

In addition to some of the unique characteristics of trust in MANETs summarized above, any trust design for MANETs should also consider how to derive a decision procedure to determine the trust of an entity, such as a trusted centralized certificate authority to take care of trust management as in wired networks. Trust may have to be determined in a highly customizable way without causing any disruption to the devices during computation and communication. A trust decision framework should not assume that all nodes are cooperative. Some nodes may act selfishly to preserve their resources. Trust should be established in a self-organized reconfigurable way to prevent service disruption due to the dynamics of MANETs. Moreover, trust in MANETs should consider the tradeoffs between security and performance such as scalability, reliability, fault tolerance, and resource consumption.

3.2.4 Related Work

Much research considered trust management. Past research focused on many aspects of trust managements such as secure routing, authentication, intrusion detection, access con-

trol, key management and trust evidence distribution and evaluation. We provide a brief summary in this section.

Secure Routing The trust based secure routing mechanism aims at detecting misbehaving nodes, both selfish and malicious ones. Marti [59] proposed a watchdog mechanism that monitors node behaviors and a pathrater that collects reputation and takes response actions. The work in Buchegger [14] determined trust levels based on self-monitored information while employing reputation collected from both direct and indirect observations and experiences. Buchegger [15] proposed CONFIDANT (Cooperation Of Nodes-Fairness In Dynamic Ad-hoc Networks) based on both direct and indirect observations to detect misbehaving nodes. Paul and Westhoff [69] proposed extended DSR (dynamic source routing) with a context-aware inference scheme to punish the accused and the malicious accuser. Michiardi [61] proposed CORE (COllaborative REputation) that has a monitoring mechanism complemented by a reputation functionality. The proposed protocol is developed to make decisions about cooperation or gradual isolation of a node. The work by He [40] proposed SORI (Secure and Objective Reputation-based Incentive). SORI encourages packet forwarding and discourages selfish behaviors based on quantified objective measures and reputation propagation by a one-way hash chain based authentication. Nekkanti and Lee [66] proposed extended AODV (Ad-hoc On-demand Distance Vector) routing using trust factors and security levels at each node. Their approach uses different levels of encryption based on the trust factors of a node to reduce the overhead. Additional work in this area includes those of [54], [34], [35], [91], [99], [100], [72], [1], [80], [85], [7], [65], [62], [75], [6], [3].

Authentication Authentication is another area that trust management research is focused on. Ngai and Lyu [67] proposed a secure public key authentication service based on their trust model to prevent propagation of false public keys in the presence of malicious nodes. Pirzada and McDonald [73] proposed a trust-based communication based on a notion of a belief, provides a dynamic measure of reliability and trustworthiness in MANETs. Weimerskirch and Thonet [93] proposed a trust model based on human behavior, noting that society can be properly considered as an ad hoc network. They used recommendations from a distributed trust model to construct trust relationships and extended it by a request for recommendations. Davis [25] proposed a reliable and structured hierarchical model for trust management in MANETs. His approach assumes that the initial certificates and public keys of all nodes are distributed by a centralized trust authority to each node before the network is deployed. Komninos, Vergados and Douligeris [46] proposed a cryptographic mechanism using a zero knowledge and challenge response protocol, which is efficient in the sense of both computational and message overhead.

Intrusion Detection There has been quite a bit of work in trust using intrusion detection systems (IDS). IDSes can help nodes to detect malicious nodes if they collaborate together. Albers [5] proposed a Local IDS (LIDS). The idea of the approach is that intrusion detection can be performed locally among trustworthy participating nodes. Ahmed [27] proposed an IDS that provides audit and monitoring capabilities that offer local security to a node and helps perceive the specific trust levels of other nodes.

Access Control Trust can also be implemented by granting access to some resources. Gray [36] proposed an integrated trust-based admission control with standard role based access control. Luo [56] proposed a ubiquitous and robust access control solution (URSA) for MANETs based on a localized group trust model so that only well behaving nodes will have access rights to network resources. Their localized group trust model for MANETs is based on threshold cryptography. A node is globally trusted only if it is individually trusted by any k trusted nodes where k is a system-wide trust threshold. Adams and Davis [2] proposed a decentralized access control system, implementing sociological trust constructs in a quantitative system to evaluate the relationships between entities. The work in [31] proposed an integrated mechanism of policy proof and reputation evolution into trust management for decision-making on access control with the goal of providing firm security as well as social security.

Key Management Virendra [90] proposed a trust-based security architecture for key management in MANETs. This architecture aims to establish keys between nodes based on their trust relationships, and to build secure distributed control using trust as a metric. Hadjichristofi [38] presented a key management framework that provides redundancy and robustness in the establishment of Security Association (SA) between pairs of nodes. The proposed key management system (KMS) adopts a modified hierarchical Public Key Infrastructure (PKI) model where nodes can dynamically take management roles. Li [52] demonstrated an on-demand, fully localized, and hop-by-hop public key management protocol for MANETs. Chang and Kou [19] proposed a Markov chain trust model to obtain the trust values (TVs) for 1-hop neighbors. They designed a trust-based hierarchical key

management scheme by selecting a certificate authority server (CA) and a backup CA with the highest TVs.

Trust Evidence Distribution and Evaluation Efforts in this area aimed at finding a general framework for trust evidence distribution and evaluation. Yan [97] proposed Personal Trusted Bubble (PTB) that considers many factors including experience statistics, data value, intrusion black list, reference (reputation/recommendation), personal preference, and PTB policies related to the entire network's security requirements. Baras [45] proposed ABED (Ant-Based trust Evidence Distribution) The idea came from the swarm intelligence paradigm, which is highly distributed and adaptive to mobility. Baras and Jiang [9] proposed a random graph theory to address distributed trust computation and establishment. Their work uses the theory of dynamic cooperative games and identifies how a phase transition from a distrusted state to a trusted state can occur in a dynamic MANETs. Theodorakopoulos and Baras [89] proposed a trust evidence evaluation scheme for MANETs. The evaluation process is modeled as a path problem in a directed graph where vertices represent entities and edges represent trust relations. Boukerche and Ren [13] proposed GRE (Generalized Reputation Evaluation) using a comprehensive computational reputation model. GRE seeks to prevent malicious nodes from entering a trusted community. Moloney and Weber [63] presented a trust-based security system that generates appropriate trust levels based on the consideration of the main characteristics of MANETs as well as context awareness. Cho [21] proposed a trust management scheme for group communication systems in MANETs. This work proposed a composite trust metric reflecting various aspects of a MANET node such as sociability and task performance capability,

and investigated the effect of the trust chain length used by a node to establish acceptable trust levels through subjective trust evaluation.

3.3 The Initial Trust Establishment Problem in SDN MANETs

As we have discussed in Chapter 1 and sections above, establishing trust in a wireless mobile ad hoc network is challenging. Even though most wireless network technologies provide choices of strong private key encryption for secure inter-node communication, the most fundamental issue of providing initial trust between nodes has been lacking, particularly for MANETs. To enable a secure SDN mobile wireless ad hoc network, it is therefore critical to design and implement an effective trust management scheme, in particular for setting up initial trust among the nodes in SDN MANETs, such as between the SDN controllers and other participants of the network.

The OpenFlow protocol specification standardizes two important aspects of the SDN data plane (1) what an OpenFlow switch hardware should support, e.g., a pipeline of flow tables, flow entry format, input and output queues, data packet processing in the switch, and so on; (2) the secure communication protocol between an SDN controller and OpenFlow switches, e.g., the message types, format, and so on. An OpenFlow secure communication session is built on top of the TLS (transport layer security) protocol. TLS and its predecessor protocol, SSL, has been widely used for supporting secure transport of information in the Internet, including being used for enabling secure web, e.g., HTTPS. TLS, by design, is a very secure protocol that deals with many types of attacks, such as replay attack. TLS/SSL supports server authentication using security certification and optionally supports

client authentication as well. However, to support certificate based authentication in TLS, a trusted third party known as a certification authority (CA) based public key infrastructure (PKI) is needed in order to combat against the man-in-the-middle attack. Specifically, a PKI certifies the true ownership (i.e., the identity) of the information contained in the certificate (such as, public keys among other information) and a chain of CAs (all the way to the root CA) that can be used to verify the signatures of CAs who ascertain the truthfulness of the information. The use and maintenance of a PKI involve complicated steps and interactions between CAs and certificate owners and require network access to the infrastructure. So far, PKI has mostly been used by clients to authenticate servers using their certificates. To support client and server mutual authentication, PKI needs to be extended to individual users.

For MANETs, it is unrealistic to constantly maintain a PKI due to various reasons, such as resource constraints, overhead, dynamics of operations (e.g., mobility of nodes, join and leave of nodes), and so on. In wired SDN networks, the availability of PKI is not an issue and its use by TLS to support secure OpenFlow protocol operations is a natural choice. However, the assumption of availability of PKI in SDN MANETs is problematic, particulary in an environment where there is no external infrastructure network support, such as WiFi or 4G networks. However, it is critical to have PKI to help establish the initial trust between the SDN controllers and the MANET nodes, and to enable mutual authentication between the controller and the nodes. Additionally, issues such as vendor equipment interoperability (while in theory, this is not an issue but can be a serious problem in practice), user groups (e.g., in military operations, soldiers form units), dynamic join and leave of nodes, needed to be taken into consideration when designing approaches for

enabling initial trust establishment.

To this end, we propose a lightweight PKI based approach. Rather than running a physical PKI within the SDN MANET or relying on the availability of a PKI that is accessible only via a supporting infrastructure, we propose to provide the equivalent PKI capabilities and services by creating a PKI with trusted virtual certificate authorities (VCAs).

3.4 Virtual Certificate Authority based Mutual Authentication for Trust Establishment in SDN MANETs

Establishing the initial trust among participating devices in an SDN based wireless mobile ad hoc network will serve as a basis for enabling ensuing secure communication of the network and for preventing various types of attacks, such as insider attack. Note that regardless of the architecture options to be adopted for a secure SDN mobile ad hoc network, we have a logically centralized controller and it is critical to enable trust through mutual authentication between the controller(s) and MANET nodes that join the network as a way of allowing keys to be distributed securely in such a system, i.e., between the controller and the nodes, and to enable subsequent secure communication.

For our SDN secure MANETs, we will enable a PKI based on using trusted virtual certificate authorities (VCAs) and therefore, relieving the MANETs of the need to rely on an external PKI. Mutual authentication using a virtual certificate authority originates from the traditional public-key infrastructure based approach for establishing initial trust among parties. Additionally, this VCA based PKI allows the SDN MANET to deal with issues such as heterogeneous networking in terms of both technology heterogeneity and device heterogeneity, more efficient management of device heterogeneity and groups of devices

under various administrative control. In the following, we discuss our approach in greater detail.

3.4.1 Components of Virtual CA Infrastructure

The virtual CA infrastructure as applied to our SDN MANETs consists of three components (1) Various types VCA devices; (2) two types virtual certificate authorities (VCAs); and (3) protocols that operate and support the VCA infrastructure.

VCA Infrastructure Devices

The VCA infrastructure has a number of devices that serve in different roles. Each type of device is responsible for specific functionality. We assume that the devices has built-in secure elements (SEs) or trusted platform modules (TPMs) for securely storage critical information for security provisioning.

- Trust Initiator (TI): is responsible for forming a secure SDN MANET, key management, key distribution and implementation of a network access control and other policies and secure mechanisms (e.g., IDS/IPS). In our SDN based MANETs, the SDN controller can serve as the TI. The TI has a certificate that is signed by the Global Virtual Certificate Authority (GVCA) (explained below) and securely stores the signed certificate as well as the certificate of the GVCA locally.

- MANET Node (MN): is the end wireless node that joins the SDN MANET. An MN could be any wireless mobile device intending to join and operate in the SDN based MANET.

- Cluster Certificate Authority (CCA): In an MANET, in practice MNs form clusters.

A cluster of MNs can naturally be under the same administrative control and/or are of the same vendor equipment, and the cluster head can serve as the CCA. For example, in a military MANET, a cluster could be formed by soldiers of the same unit and under the command of a commanding officer (CO). The CO can then act as the CCA that can help certify/vouch for the identities of MNs in the cluster if needed. In the context of the proposed VCA infrastructure, a CCA will act as the trusted third party between the MN and the TI (the details of how this works will be explained later). There might be more than one cluster and CCA in an MANET.

- Cluster Virtual Certificate Authority (CVCA): This is a virtual entity for a cluster of MNs including the CCA. A CVCA is unique to a specific cluster and is the trusted third party for the CCA and the MNs in the cluster. It is responsible for signing the certificates of the MNs and the CCA prior to device deployment. The signed certificates are securely stored in the devices (e.g., SEs or TPMs). In addition, the certificate of the CVCA is also stored securely on CCA and MNs upon signing their respectively certificates. This CVCA is essentially the root-CA for these signed certificates and its private key is kept secure outside of the MANET and not revealed to the CCA or MNs of the cluster.

- The Global Virtual Certificate Authority (GVCA): Similar to CVCA, the GVCA is also a virtual entity and is unique to the MANET. The GVCA is the trusted third party between the TI and the CCA(s). It is responsible for signing the certificates of the TI and the CCA(s) prior to deployment. The signed certificates are securely stored in the devices (e.g., SEs or TPMs of the TI and the CCA(s)). In addition, the

certificate of the GVCA is also stored securely on CCA(s) and the TI upon signing their respectively certificates. This GVCA is essentially the root-CA for these signed certificates and its private key is kept secure outside of the MANET and not revealed to the CCA(s) or the TI of the MANET.

Virtual Certificate Authority (VCA)

In the proposed VCA infrastructure, there are two types of root CAs. Both types are virtual root CAs as referred to above. The VCA(s) are not physical entities that can be referenced, but abstract concepts. However, VCAs are considered to be fully functional root CAs with their own addresses, private-public key pairs and their own certificates. The main role of a VCA is verification and the signing of other devices' certificates. The VCA can act as the trusted third party to verify devices and act as a basis for initial trust. The two types of defined VCAs are further elaborated as follows: VCA.

- Cluster Virtual Certificate Authority (CVCA): provides a mechanism by which a MANET node (MN) and a cluster's Certificate Authority (CCA) can verify each other's certificates before mutual authentication. The cluster administrator signs certificates of the CCA and MNs of the cluster before implanting them on each MN and the CCA ahead of deployment using the private key of the CVCA. Both the CCA and the MN can verify each other's certificate by using the CVCA's public key. Additionally, the certificate of the CVCA is also securely stored on the MNs and the CCA.

- The Global Virtual Certificate Authority (GVCA): provides a mechanism for different devices to mutually authenticate, by providing a basis for this initial trust. The

foundation for initial trust is the private key of the GVCA. The TI and every CCA has a certificate signed by the GVCA. The MANET administrator signs certificates of the CCA(s) and the TI of the MANET before implanting them on each CCA and the TI ahead of deployment using the private key of the GVCA. The private key of the GVCA is off the devices and off the MANET.

While there are two types of root CAs in the VCA infrastructure, the GVCA has the highest root level since it is the basis for initial trust between CCAs and the TI. The CVCA is also a root CA, but it is a level-2 root CA. It is the basis for initial trust between MNs and the corresponding CCA. The idea of two different root level CAs allows for a high degree of interoperability in dealing with different vendor equipment while giving each individual cluster a high degree of control and management. This facilitates our SDN based initial trust management approach to deal with device heterogeneity.

3.4.2 VCA Infrastructure Functionality

The basic functionality that a node is required to have in order to join in an SDN based MANET with the proposed VCA infrastructure is discussed in this subsection. The VCA infrastructure functionality includes: a device (MN, CCA) should be able to request a certificate, verify the signature on a certificate as well as participate in a challenge and response procedure, and a CCA should be able to sign a certificate. For a CCA to sign a certificate, the CCA needs to secure store its private key locally in its SE or TPM.

- Requesting a Certificate: An MN or a CCA can initiate a request for a certificate from the TI to join the SDN MANET.

- Verifying a Certificate: The verification of a certificate implies that the data on the certificate presented is correct and can be trusted. The verification process involves using the CVCA's or GVCA's public key or a previously authenticated CCA's public key to verify the certificate.

- Challenge and Response: Once a certificated is verified, a followup challenge and response procedure is necessary to protect against a replay attack. For example, the certificate is replayed certificate that belongs to another device. For example, the certificate can be obtained via eavesdropping. Therefore, a challenge and response technique such as public key encryption, private key decryption of nonces, can be implemented to verify the true ownership.

- Signing a Certificate: When requested by a MN for signing its certificate, the CCA must first authenticate the MN before signing its certificate. Upon the successful authentication of the MN, the CCA can then sign the certificate presented by the MN after removing its original signature.

3.4.3 Device Mutual Authentication

The VCA infrastructure supports two devices that have no prior knowledge of each other to perform secure authentication (e.g., between the TI and a CCA). For devices (CCA(s) and MNs) to mutually authenticate with the TI (i.e., the SDN controller), two steps have to take place. First, a CCA of a cluster mutually authenticated with the TI and joins the SDN MANET after successful mutual authentication. Second, an MN of the same cluster as the CCA that has joined the SDN MANET mutually authenticates with the TI through the help

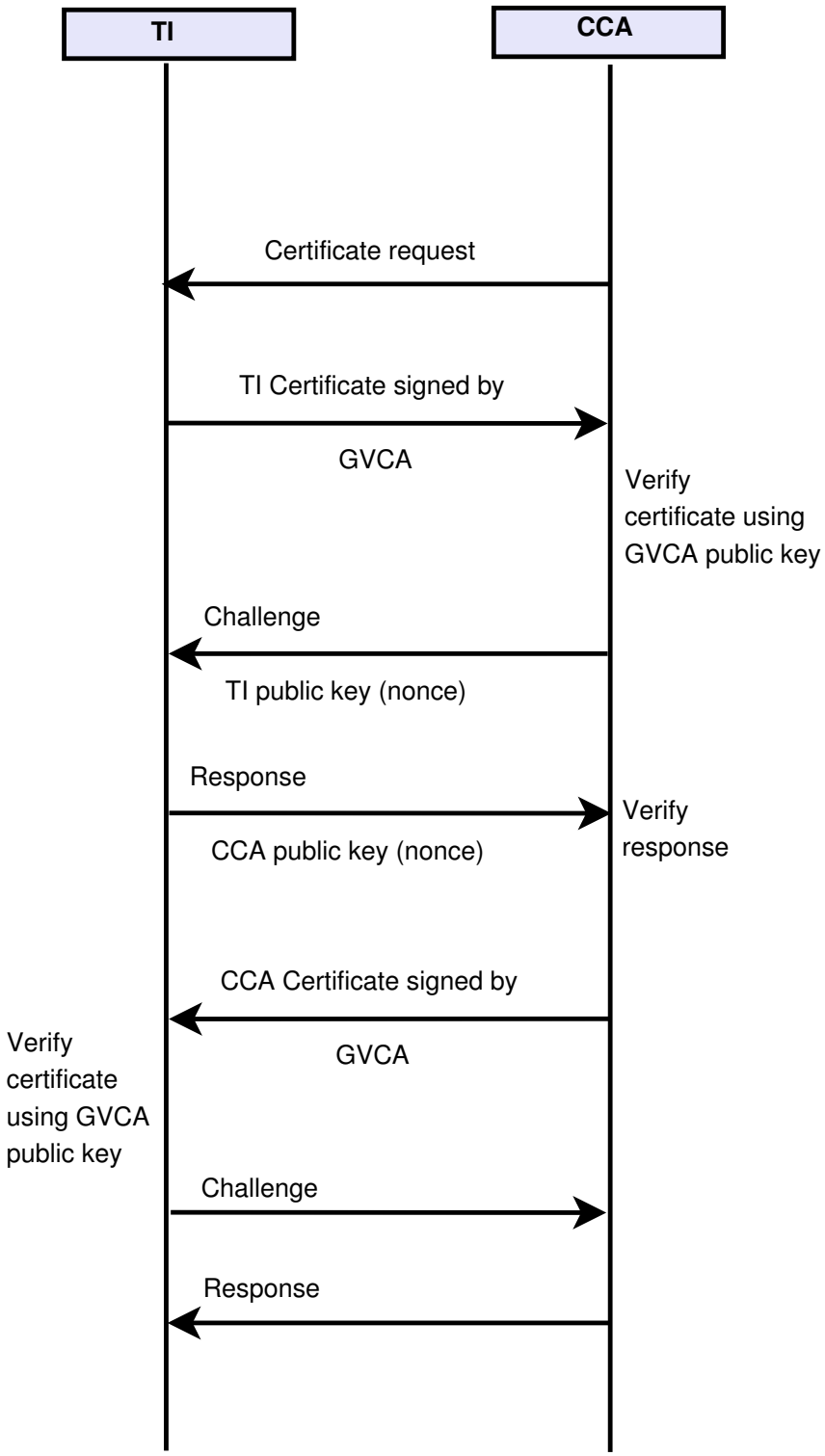

Figure 3.1: Authentication between the TI and a CCA using Virtual Certificate Authorities.

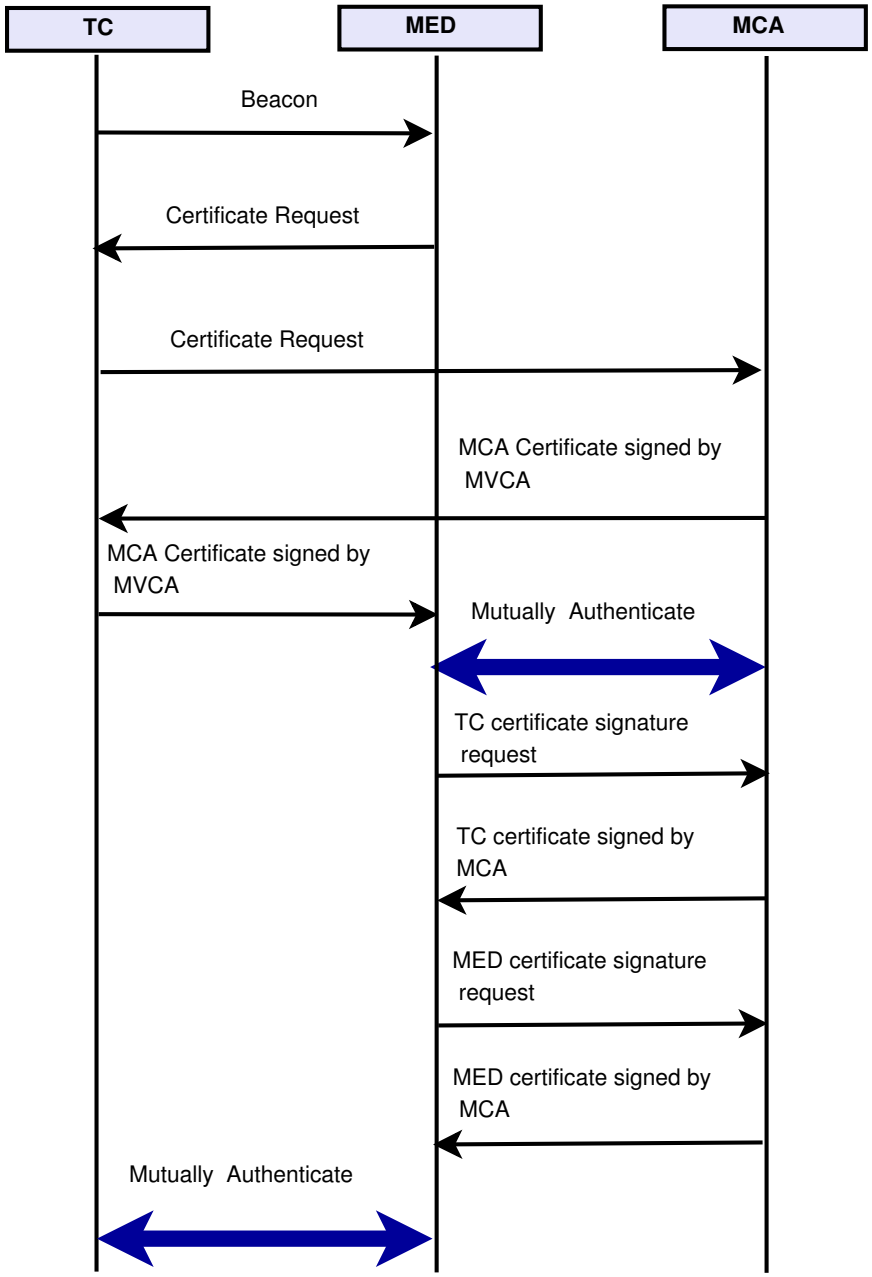

Figure 3.2: Authentication between MNs and TI using Virtual Certificate Authorities.

of the CCA.

Step I: CCA and TI Mutual Authentication

A CCA node starts out by contacting and requesting the certificate of the TI which responds by sending its certificate signed by the GVCA. Upon receipt of the signed certificate from the TI, the CCA can verify the certificate using the public key of the GVCA which the CCA obtains from the certificate of the GVCA stored locally. To ensure that the certificate from the TI is not replayed, the CCA will conduct a challenge and response test by sending a nonce encrypted using the TI's public key and waiting for a response from the TI with the same nonce encrypted using the CCA's public key. The CCA decrypts the received nonce and compares with the nonce it sent. If the two nonces match, then the TI is the true owner of the certificate and its identity has been authenticated. The authentication of the CCA by the TI can be accomplished in a similar manner, after which the CCA will join the SDN MANET. The flow of the Step I authentication is shown in Figure 3.1.

Step II: MNs and TI Mutual Authentication

The TI and MNs authentication procedure is shown in Figure 3.2. The steps involved are described as follows.

- An CCA authenticates the TI and requests to associate to the network.
- The TI authenticates the CCA and authorizes it to associate.
- The MN issues a pre-authentication request to the TI for a certificate of a CCA device and CVCA.
- The TI does not have this certificate. It has previously authenticated a CCA from the same cluster. It requests this certificate from the CCA.

- This certificate has been implanted on the CCA. It forwards it to the TI.
- The TI in turn forwards this to the unauthenticated MN.
- The MN authenticates the CCA.
- The MN then requests the TI's certificate signed by the CCA.
- The MN can now authenticate the TI before requesting to associate. The MN provides its certificate signed by the CVCA as part of the association request.
- The TI can request the CCA to verify the certificate before authenticating the MN.
- Having completed authentication the TI can authorize the MN to associate.
- Secure session keys between network peers can be optionally obtained as done in hybrid cryptosystems.

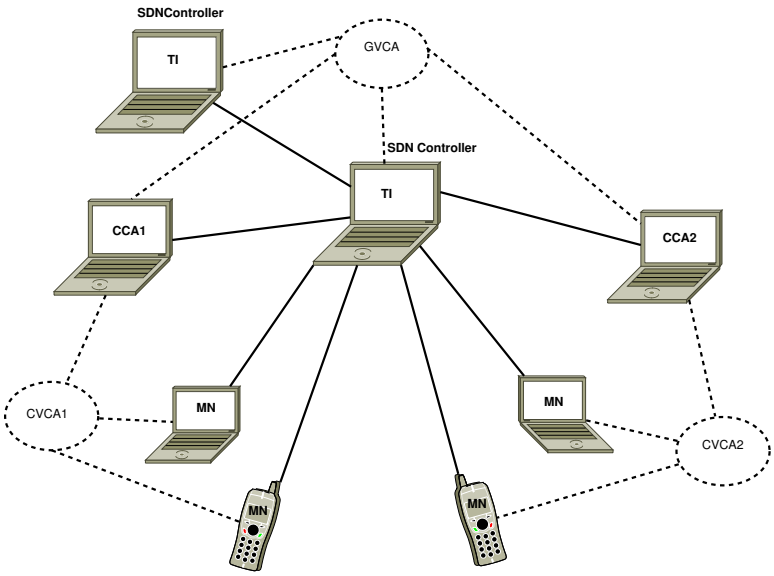

Figure 3.3: Integration of VCA infrastructure with SDN based MANETs for centralized controller with controller replication.

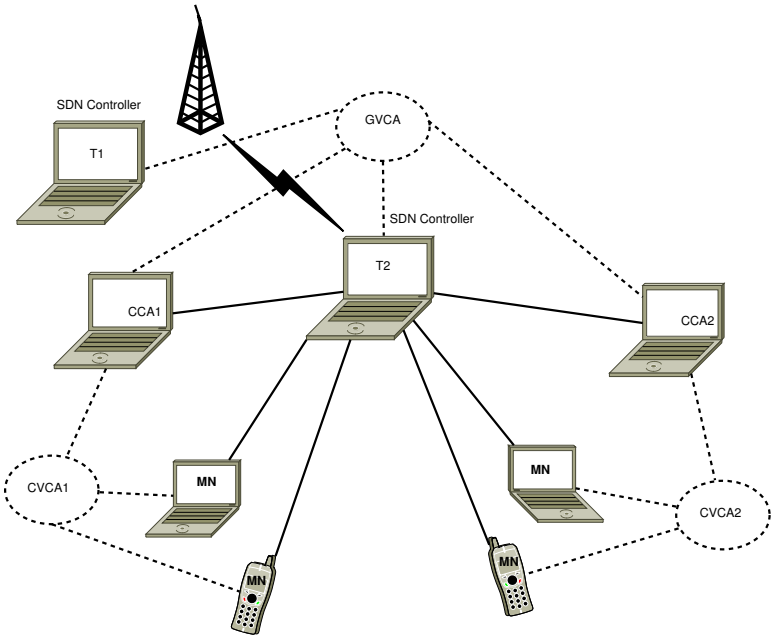

Figure 3.4: Integration of VCA infrastructure with SDN based MANETs for centralized controller with controller replication and external infrastructure network.

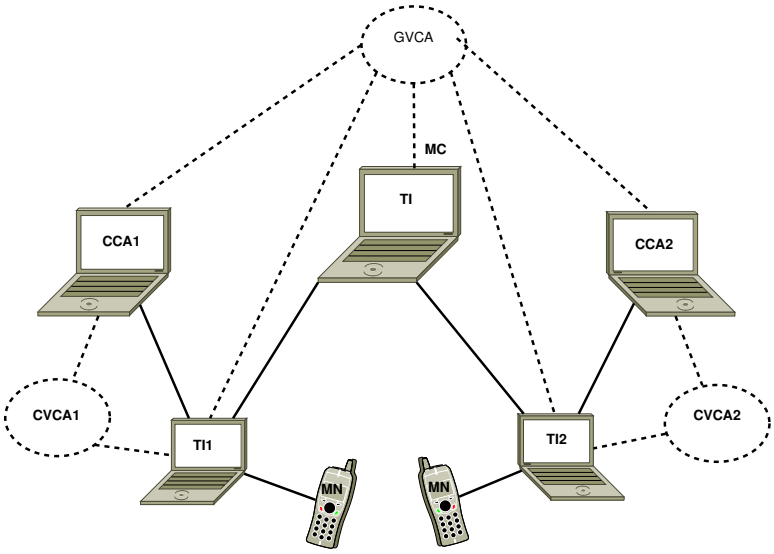

Figure 3.5: Integration of VCA infrastructure with SDN based MANETs for hierarchically distributed architecture.

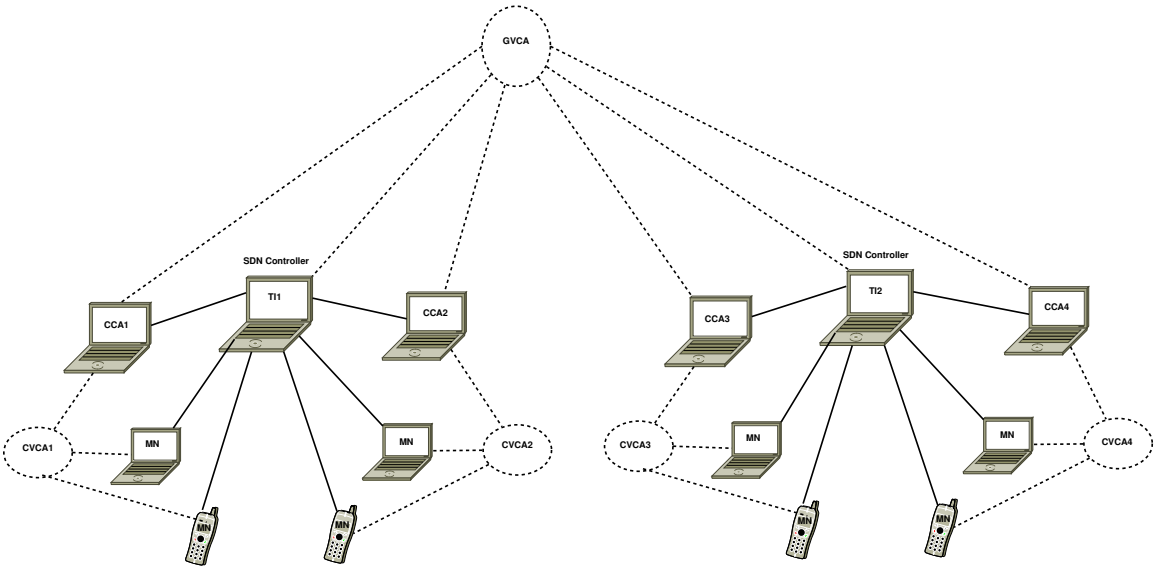

Figure 3.6: Integration of VCA infrastructure with SDN based MANETs for fully distributed architecture.

3.4.4 Integration of VCA Infrastructure with SDN based MANETs

Note that the VCA infrastructure is not limited to a simple star topology and it works equally well in general network topology with slight extension of the OpenFlow. The CCA(s) and MNs in a mesh or cluster tree network can join a network without being in direct communication with the TI because traffic can be routed to the TI on behalf of the nodes. Having presented the detail of the proposed VCA infrastructure for initial trust management, we now examine how the proposed VCA infrastructure can be fitted into the different SDN based secure MANET architecture design options of Chapter 2.

For design option 1 (D1) where the SDN MANET has a controller with replication, the VCA infrastructure can be deployed seeminglessly (Figure 3.3). As a controller is the core the MANET, the TI will naturally reside in the controller and authentication related information will be replicated to other controllers. The TI and its replica will be properly configured with certificates signed by the GVCA and store the certificate of the GVCA.

The cluster head of a cluster of MNs will assume the role of a CCA. The administrator of a cluster will configure the MNs and the CCA with certificates signed by the CVCA and the certificate of the CVCA. Additionally, the CCA will have its certificate signed by the GVCA and store the certificate of the GVCA as well. MN mobility can be supported without any problem. A new MN can be mutually authenticated before being allowed to join the network.

For design option 2 (D2) where the SDN MANET has a controller with replication and external network infrastructure support, device authentication has at least two options. The first option is to rely on the traditional well established PKI as in this case, the external network infrastructure offloads the burden of maintaining the PKI from the SDN MANET. The second option is to integrate the proposed VCA infrastructure and use the VCA infrastructure only. In this case, the configuration, running and usage of the proposed VCA infrastructure is similar to that of design option 1 (D1) (Figure 3.4).

For design option 3 (D3) where the SDN controllers are divided into a master controller (MC) and a few secondary controllers (SCs), the CCA and MNs of a cluster will authenticate and join a secondary controller where a TI resides (Figure 3.5). The MC will also run as a TI and authenticate with the secondary controllers. All the controllers will be configured with their own certificates signed by the GVCA and store the certificate of the GVCA. The configuration of CCA(s) and MNs with proper certificate information will be performed in a way similar to D1 and D2. When a new MN attempts to join the MANET, it needs to authenticate a TI and gets authenticated by a TI. The preferred TI will be the one where the new MN's CCA is associated with. In this case, the authentication can proceed in a straightforward manner. If the new MN cannot communicate directly with the preferred

TI, the certificate request message will be forwarded to the preferred TI (and the CCA) and the closest TI simultaneously. Then the MN can perform the mutual authentication with the the closest TI with the help of the preferred TI and the CCA. When an MN moves and attempts to switch from one SC to another (i.e., resulting in changing association from one TI to another), two options are available. The first option is no further authentication is necessary for the MN to be associated with a new SC/TI because the MN has been authenticated previously and is part of the MANET. However, the new SC/TI with which the MN will be associated will contact the MC/TI which in turn may contact the MN's original SC/TI to verify that the MN has indeed been authenticated. The second option is force the moving MN to undergo a new round of mutual authentication with an SC/TI. The steps taken will be the same as that of a new MN that attempts to join the MANET.

For design option 4 (D4) where the SDN controllers are fully distributed and each controller is responsible for a subset of the nodes, the roles of TIs will again naturally fall into these controllers (Figure 3.6). The TIs will be properly configured with certificates signed by the GVCA and store the certificates of the GVCA. So are the CCA(s) and MNs before deployment. Similar to D3, when a new MN attempts to join the MANET, it needs to obtain mutual authentication with a TI. Depending on where the new MN is located, the MN may be able to mutually authenticate with the local TI if its CCA is associated with the TI. If the CCA of the new MN is not associated with the local TI, then the MN will obtain mutual authentication via the local TI to obtain the certificate of the CCA that moved. The steps that follow are similar to those in the case of D3. Likewise, when an MN moves and attempts to associate with a new TI, the MN has two options as described in D3.

3.4.5 Advantages of VCA Infrastructure

The proposed VCA infrastructure solves the issue of initial trust by implementing a flexible and cost-effective PKI in an SDN MANET (Figures 3.3, 3.4, 3.5, 3.6). The VCA approach provides many advantages to the SDN based wireless networks, such as

- The private key(s) of the VCA(s) are tamper-proof as they are not stored on any device.
- Limited overhead because only the certificate of the device itself and the certificate of the signer are implanted prior to deployment.
- Flexibility, properly configured new devices after the MANET deployment can still authenticate themselves to the devices already deployed.
- Interoperability among devices of different vendors is facilitated.
- Individual clusters/groups/vendor specific devices can manage access control policies.

3.4.6 Potential Disadvantages of VCA Infrastructure

The proposed VCA infrastructure shows that this lightweight approach helps implement an attack resilient framework. However the infrastructure is still not attach proof. One possible attack is denial of service in nature. An unauthenticated node attempting to join the MANET can continuously and repeatedly send legitimate requests to authenticate, therefore overwhelming the TI. We plan to conduct a full review and analysis of other potential

security weaknesses and provide countermeasures, such as integrating with an IDS/IPs system.

3.5 Summary

In this chapter, we have provided a taxonomy of different types of trust management framework in MANETs. Past research has been summarized, focusing on many aspects of trust managements such as secure routing, authentication, intrusion detection, access control, key management, trust evidence distribution and evaluation. We presented the challenges of designing a trust management framework that is suitable for mobile ad hoc wireless networks.

We addressed the most fundamental issue of providing initial trust between nodes in MANETs. To enable a secure SDN mobile wireless ad hoc network, in particular for setting up initial trust among the nodes in SDN MANETs, such as between the SDN controllers and other network nodes, we have proposed the design of a lightweight PKI. Rather than running a physical PKI within the SDN MANET or relying on the availability of a PKI that is accessible only via a supporting infrastructure, we proposed to provide the equivalent PKI capabilities and services by creating a PKI with trusted virtual certificate authorities (VCAs). We have examined the integration of the VCA infrastructure with the different design options of the SDN based MANET architecture.

Chapter 4

Intrusion Detection and Prevention in SDN based MANETs

4.1 Introduction

Network Intrusion Detection Systems(IDSes) are hardware and/or software security systems that monitor the networked systems (e.g., user devices and networking equipment), networking software and/or applications for malicious activities or policy violations. The functions of the intrusion detection system are to detect possible incidents and anomalies with the aim of determining and locating malicious activities before they do real damages to the network and users. There are network based (NIDS) and host based (HIDS) intrusion detection systems. NIDS is a network security system focusing on the attacks on the network IDSes strive to identify malicious activities, log information about and report these activities, and optionally attempt to block/stop these identified malicious activities. An intrusion detection system with an explicit prevention component offers a combination IDS and IPS (intrusion prevention system) technologies that are capable of working solely in the IDS watch-and-alert mode or being fully deployed to also stop attacks in the IPS mode.

Network Intrusion Detection Systems (NIDS) are placed at a strategic point or points

within the network to monitor traffic to and from all devices on the network. Once an attack is identified, or abnormal behavior is sensed, the alert can be sent to the administrator. An intrusion detection system (IDS) differs from a firewall in that a firewall examines incoming and/outgoing network traffic for intrusions in order to stop them from happening. Firewalls conducts access controls on traffic to prevent intrusion and do not signal an attack that occurs inside the network. However, an IDS evaluates a suspected intrusion once it has taken place and signals an alarm. An IDS may also watch for insider attacks though this form of attack is much more difficult to detect and prevent. An IDS uses one of two detection techniques: (1) Signature-based (or Misuse-based) IDS that detects abnormal activities/-software (e.g., virus, malware, etc) based on comparing the characteristics of suspected activities against a database of signatures of known attacks. (2) Statistical anomaly-based IDS that learns a baseline normal behavior of a system and sounds an alarm whenever the system behavior deviates from the baseline significantly. Signature-based IDS systems have high detection precision and low rates false alarms, but cannot detection unknown attacks while anomaly-based IDS systems often generate a large quantity of alerts and many of these alerts are false positives. Anomaly-based IDS systems have the advantages of detecting unseen attacks if properly designed. In practice, however, anomaly-based IDS systems are rarely used alone as they produce too many alarms to cause "alarm fatigue" and therefore often used in combination with signature-based IDS systems. With the advance in machine learning, particularly in recent big data and deep learning neural network technologies, anomaly-based IDS systems with human-in-the-loop have the potential to go far beyond what they are capable of today.

We focus on examining various ideas of integrating IDS techniques within SDN based

MANETs and the architecture design options of Chapter 2, building upon past research on IDS/IPS in MANETs, and explore the advantages and overcome the weaknesses of the centralized control plane in the context of SDN MANETs. The rest of the chapter is organized as follows. In Section 4.2 we briefly present the IDS/IPS security capabilities, in Section 4.4 related work and software defined intrusion detection and prevention systems architecture are described, and we analyze several design options in Section 4.5 for software defined intrusion detection and prevention systems in SDN based MANETs.

4.2 Network IDS/IPS System Security Capabilities

IDS/IPS systems provide a wide variety of security capabilities [79]. These capabilities can be divided into four categories: information gathering, logging, detection, and prevention.

- Information Gathering: Information gathering capabilities are to collect information on hosts or networks from observed activities and to identify general characteristics of the network.

- Logging: IDS/IPSes typically perform extensive logging of data related to suspected incidents and system events. The data can be used to confirm the validity of alerts, investigate incidents, and correlate events between the IDS/IPS systems and information from other sources.

- Detection: The technologies of IDS/IPS systems offer extensive, broad detection capabilities. Detection accuracy and recall can be improved from the default configuration by tuning and customization capabilities of IDS/IPS technologies. A few examples of such capabilities include

  - Thresholds. A threshold is a value that sets the limit between normal and abnormal behavior.
  - Blacklists and Whitelists. Blacklists, known as hot lists, are typically used to allow IDS/IPSes to recognize and block activities that are highly likely to be malicious. A whitelist is a list of discrete entities that are known to be benign.
  - Alert Settings. Most IDS/IPS technologies allow to customize alert types to prevent the IDS/IPS from being overwhelmed by alerts.
  - Code Viewing and Editing. IDS/IPS technologies may provide code viewing and editing capabilities to ensure that tuning and customizations are accurate.

- Prevention: IDS/IPSes offer multiple prevention capabilities. The specific capabilities vary by IDS/IPS technology types. For example IDS/IPS sensors may have a learning or simulation mode that suppresses all prevention actions and instead indicates when a prevention action would have been performed. This allows to monitor and fine-tune the configuration of the prevention.

4.3 IDS/IPS Challenges in Mobile Ad Hoc Wireless Networks

As IDS/IPSes present a second line of defense in MANETs, significant challenges exist [98] due in part to the unique characteristics of mobile ad hoc wireless networks, such as:

- Often with no fixed infrastructure;
- Peer-to-peer architecture with multi-hop routing;

- Physical vulnerability of mobile devices;
- Stringent resource constraints in many cases;
- Wireless communication medium, and
- Node mobility.

Effective security solutions must meet some stringent criteria in order to be suitable for operating in an ad hoc wireless network environment. In addition, the security solutions should be lightweight, decentralized, reactive and fault-tolerant.

4.4 Software Defined Intrusion Detection and Prevention Systems

As discussed in Chapter 1, the software defined networking paradigm offers layered, modular, and logically centralized control of a network by separating the control plane from the data plane. Taking advantages of the advance in the high performance generic computing platform such as low-cost and highly reliable computing clusters and server farms as well as software technologies in hypervisors for virtual machines and virtual network switches, SDN greatly facilitates the implementation of network virtualization and network function virtualization (NFV). As a result, many network control and management functions and services can be provided at centralized locations, replacing many customized "middleboxes". Software defined intrusion detection and prevention systems make monitoring large scale networks and offer incidents detection/prevention easier than ever before. Recent years have witnessed some research that focuses on using the software defined network

framework in dealing with the complexity and inflexibility of security provisioning in the traditional networks.

Ethane [18] proposes an SDN architecture that allows managers to enforce fine-grained access control policies in enterprise networks. AVANT-GUARD [83] addresses two security challenges in SDN-enabled networks: (1) secure the interface between the control plane and data plane through a connection migration technique to protect the control plane from the saturation attacks. (2) Speed up the responses to threats. FRESCO [82] provides a development environment and a resource controller to facilitate the design and composition of SDN-enabled security applications. FRESCO targets at the general networking environments and does not consider the features of mobile environments. OpenSAFE [8] leverages SDN open networking technology such as OpenFlow for directing spanned network traffic in arbitrary ways to many service points for traffic security monitoring and filtering at line rates. The work does not attempt to provide a comprehensive detection and prevention solution. In OpenFlow Random Host Mutation (OFRHM) [44], the OpenFlow controller frequently assigns each host a random virtual IP that is translated to/from the real IP of the host. This mechanism can effectively defend against stealthy scanning, worm propagation, and other scanning-based attacks, but does not work when the attackers know the internal addresses of the victims. SDN-based IDS/IPS solutions of [23] deploy attack graphs to dynamically generate appropriate countermeasures to enable the IDS/IPS in a cloud environment. SnortFlow [94] focuses on the design and preliminary evaluation of OpenFlow enabled IPS in a cloud environment. The work of [95] proposes a new IDPS architecture (a SDN-based IPS (SDNIPS)) based on Snort IDS and Open vSwitch (OVS) also for intrusion detection and prevention in the cloud. A distributed elastic intrusion detection architecture

(DEIDtect) [81] exploits the increasing deployment of cloud computing and software defined networking technology in enterprise and campus environments to deal with current inflexibilities associated with compute and network resources required by security tools. Learning Intrusion Detection System (L-IDS) [84] is a network security service for protecting embedded mobile devices within institutional boundaries which can be deployed alongside existing security systems with no modifications to the embedded devices. L-IDS utilizes the OpenFlow software defined networking architecture, which allows it to both detect and respond to attacks as they happen.

However, much of the past research work lacked a systematic examination of SDN security issues in MANETs. For example, (1) how to design an efficient SDN-based IDS/IPS solution with effective and scalable components in an MANETs, taking into account unique characteristics in such environments and subjecting to constraints of such networks; (2) how to integrate such solutions with various types of SDN-based MANET architecture to provide a dynamic defensive mechanism against ever evolving forms of attacks. In the sections that follow, we will present a comprehensive examination of SDN IDS/IPS for MANETs and discuss the integration with our proposed SDN MANET architecture design options of Chapter 2.

4.5 Software Defined Intrusion Detection and Prevention in MANETs

4.5.1 Security Threats

While we have briefly discussed security weaknesses of SDN based MANETs in Chapter 1, a more detailed description of potential security threats in SDN MANETs is called

for. We present these threats from two different perspectives. Table 4.1 shows threat vectors from the network infrastructure point of view. A total of seven possible threat vectors are listed [48]. Very briefly, (V1) forged or faked traffic flows can be used to attack (DoS) switches and controllers; (V2) attacks on vulnerabilities in switches can drop or slow down packets, clone or deviate network traffic (e.g., for data theft purposes), or even inject traffic or forged requests to overload the controller or neighboring switches; (V3) attacks on control plane communications can be used to generate DoS attacks or for data theft, taking advantages of weaknesses of TLS/SSL; (V4) attacks on vulnerabilities in controllers may lead to the compromise of the SDN controller and the entire network; (V5) lack of trust between the controller and management applications may lead to DoS attacks and/or data theft; (V6) attacks on vulnerabilities in administrative stations can lead to hijacking and reprogramming of the entire network; and (V7) lack of trusted resources for forensics and remediation can compromise diagnosis and recovery.

Table 4.2 shows different types of security threats in a SDN enabled wireless mobile networks from the layered SDN abstraction point of view [55]. The table is self-explanatory, listing the different types of threat that can target different layers in the SDN architecture and a short description of the threat and reasons. Different threats can target the data plane, the control plane, the control application plane, and the interfaces between the data plane and the control plane as well as between the control plane and the application plane.

Threat Vector	Specific to SDN?	Impact on general SDN Networks	Impact on SDN MANETs
(V1) Forged or faked traffic flows	No	Can a door for DoS attacks	Augmented impact
(V2) Attacks on vulnerabilities in switches	No	Potentially augmented impact	Augmented impact
(V3) Attacks on control plane communications	Yes	Exploit communication with SDN controller	Augmented impact
(V4) Attacks on vulnerabilities in controllers	Yes	Controlling SDN controller compromises network	Augmented impact
(V5) Lack of mechanisms to ensure trust between controller and management applications	Yes	Malicious apps can be deployed on controller	Augmented impact
(V6) Attacks on vulnerabilities in administrative stations	No	Potential augmented impact	Augmented impact
(V7) Lack of trusted resources for forensics and remediation	No	Important for diagnosis and recovery	Augmented impact

Table 4.1: Threat Vectors in SDN Networks and Their Impact.

4.5.2 A Taxonomy of SDN IDS/IPS Mechanisms for MANETs

We present a taxonomy of SDN IDS/IPS mechanisms for MANETs based on where the mechanisms take effects to deal with threats. Software defined networking facilitates the design and implementation of IDS/IPS mechanisms by providing a global network view and flexibly means to direct, mirror, and inspect the network traffic and taking proper actions. Taking advantages of the flexibility of software defined networking, intrusion detection/prevention operations can be carried out in a number of modes:

- On-path Detection: A network flow is a sequence of packets sent from a source to a destination along a route. The IDS is placed on the packets route so that every packet goes to the IDS to be analyzed and then forwarded to its destination. SDN switches route traffic by matching flows against rules held in a pipeline of flowtables with entries of rules. Each rule has an associated set of actions to take on all matched

SDN MANET Layer	Type of Threat	Threat Reason and Description
Application	Lack of authentication and authorization	Possible huge number of third party apps
	Fraudulent rules insertion	False flow rules generated by Malicious applications
	Access control and accountability	Lack of binding mechanisms for applications
Control	Dos, DDoS, controller hijacking or compromise	Visible nature of control plane
	Unauthorized controller access	No compelling mechanisms for enforcing access ctrl on backhaul devices
	Scalability or availability	Centralized intelligence
Data	Fraudulent flow rules	Lack of intelligence
	Flooding attacks	Limited capacity of flow tables
	Controller and DP switch masquerading	Lack of strong authentication
Control-Data Interface	TCP-level attacks	TLS is responsible to TCP level attacks
	Man-in-the middle attack	Optional use of TLS and complexity in configuration of TLS
Application-Control Interface	Illegal controller access, policy manipulation and fraudulent rule insertion	Limited secure APIs, lack of binding mechanisms b/w apps and controller

Table 4.2: SDN Security Threats: A Layered Perspective.

packets. The OpenFlow specification currently defines 15 packet header fields which can be matched against the rules. Whenever a rule is triggered, the switch updates a set of traffic counters for detailed measurement of network activities. If an incoming packet does not match any flowtable rule, it is forwarded to the controller. The controller will decide the required actions. However, this on-path IDS architecture may lead to poor network performance because every packet must be analyzed before being forwarded. Therefore, it is not directly applicable in SDN MANETs.

- Off-path Detection: In this approach the IDS is a separate node of the network connected to the controller, and every packet that arrives is mirrored to the port in which the IDS is connected. If an anomaly is detected by the IDS, an alert will be sent to the controller to take proper actions. This architecture may not affect the performance in a wire network but the MANET performance may suffer. Additionally, packets of certain threats it may go to the destination before the threat is identified as malicious. This architecture is also not directly applicable to MANETs especially for infrastructure-less MANETs due to node mobility, limited resources, dynamic node join and leave.

- Hybrid Detection: This approach combines on-path and off-path detection with functions of IDS split into multiple entities and conducted collaboratively among the entities located on both the data plane and the control plane. In theory, this architecture will be more scalable. With proper design and balance of functions and load, this architecture may be applicable to MANETs.

4.5.3 Proposed SDN MANETs IDS/IPS Architecture and Components

We propose a distributed cross-layer (e.g., data plane, control plane, and control application plane) approach to developing, deploying, and integrating IDS/IPS in SDN MANETs. We will also elaborate some component mechanisms that make this approach scalable, resilient, extensible, and responsive.

Application plane

The architecture uses a layered approach and integrates component mechanisms across different layers 4.1. At the top is the control application for IDS/IPS. The application is an finite state machine (FSM) based IDS/IPS control application developed using high-level SDN programming languages such as Pyretic/Frenetic to allow developers to easily express the intention, objectives, and security policies. The high-level programming languages are then translated into OpenFlow messages that ensure the flowtable rules are consistent and avoid race conditions. Additionally, this also allows the application to be easily extensible to integrate new mechanisms and new security policies (either stipulated or automatically learned). The IDS/IPS control application can be further enhanced with machine learning capabilities built in, therefore allowing new rules and policies to be learned over time and translated to low-level messages to reprogram the switch flowtables.

Control Plane

It is well know that off-the-shelf operating systems, from different families, have few intersecting vulnerabilities, which means that OS diversity constrains the overall effect of attacks on common vulnerabilities. Therefore, to prevent controller saturation, the proposed architecture uses multiple controllers, diverse controllers, diverse network operating

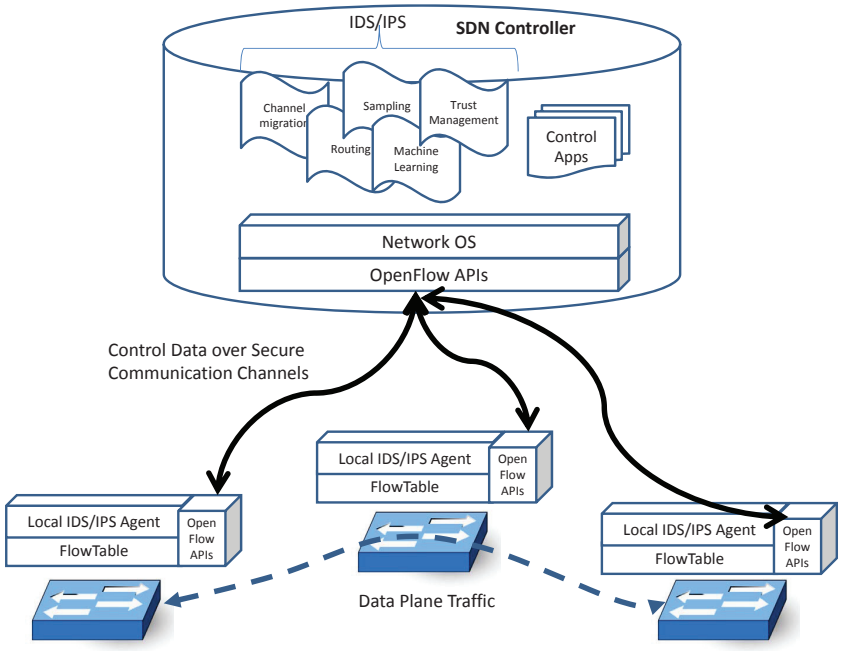

Figure 4.1: Hybrid IDS/IPS for Software Defined MANETs.

system (NOS) and protocol for communication. This is in fact consistent with our proposed SDN MANET architecture design options of Chapter 1 where we propose to use multiple controllers for better resilience.

Trust between devices and controllers. Establishing trust between devices and controllers is an important requirement for overall control plane trustworthiness. Network devices should be allowed to associate with controllers dynamically but without incurring in less reliable relationships. In the proposed architecture, the southbound interface/communication between the control plane and the data plane is enhanced using our proposed improved trust management scheme (Chapter 3) for implementing assured mutual authentication to combat against threats on the communication channel. After authentication, a simple approach that can be implemented would be to have authenticated white lists of

known trusted devices kept at controllers. The interface between the control plane and data plane can be additionally secured through a connection migration technique and multipath control message routing to protect the control plane from the saturation attacks. Furthermore, a rate limit can be imposed on flow level control messages sent to the controller to limit the use of SDN resources. Similar limits can be imposed on the use of other resources such as flowtables and CPU processing. These multiple lines of defense will enable the proposed architecture to deal with problems such as DoS by the legitimate request messages for SDN controller certificate. On the other hand, a switch can be allowed to dynamically associate with several controllers in a secure way (e.g., by using threshold cryptography to detect malicious controllers and authentication, which would hinder man-in-the-middle attacks, for instance). A switch associated with different controllers would be able to automatically tolerate faults. Other advantages include increasing control plane throughput, load balancing, and reducing control delay.

Data Plane

To promote the scaling of IDS, the system needs to be lightweight and leverage scaling opportunities. For example, the architecture splits on-path detection responsibilities over switches and aggregates intermediate results in order to split the expensive IDS processing to form a logically centralized IDS/IPS system. The SDN controller uses the OpenFlow protocol to fully and flexibly control network data flows for scalable network traffic inspection. The SDN controller maintains flow information useful for security purposes such as source and destination MAC addresses, port numbers, and traffic statistics. A distributed traffic sampling at network switches for malicious traffic inspection will be used to increase the scalability of IDS/IPS. The inspection capacity of the IDS can be defined as the

maximum amount of traffic that the IDS can process without significant degradation of inspection performance. Because this capacity is limited, the IDS cannot inspect all traffic on the network. Instead, it is desirable to sample traffic flows that are most likely to contain malicious packets. The sampling rates can be adjusted at network switches to minimize the probability that malicious packets are undetected in traffic samples forwarded to the IDS. As a result, a sampling extension for OpenFlow is needed to promote the development of security applications such as monitoring. To overcome the limitation of existing OpenFlow mechanisms to access packet-level information such as port manipulation and mirroring, the controller needs to be able to specify which packets shall be sampled and what part of packets shall be selected, and to where it shall be sent. Packets can be sampled either stochastically or deterministically and thus making it flexible to meet the needs of different applications. The design needs to be implemented on data plane and takes into account the optimization and the dynamics of the mobile environment.

To improve the IDS/IPS responsiveness so that security applications can efficiently access network statistics (on the data plane) to respond to threats, a self-healing mechanism can be instituted into the switches to create triggers that can be inserted by the control plane to register asynchronous callback and add conditional flowtable rules that are activated when a predefined trigger condition is detected. This mechanism is supported the new characteristics of SDN paradigm with the introduction of self-healing mechanisms whereby conditional rules. These rules are installed on the switches by the control plane and are activated once a certain condition is met. The condition is usually related to the switchs collected statistics, such as when the number of packets belonging to a certain flow received during a certain period of time exceeds a predefined threshold. The activated

rule specifies how the switch should respond when the specified condition is met. These reactions provide automated resiliency against attacks. The reaction specified by the conditional rule could be to drop the packets specified by the rule or to forward those packets through different paths to mitigate the load on certain parts of the network. This provides resiliency against denial of service (DoS) attacks targeting network hosts or network links. The reaction could be to modify the destination address of certain packets so they are delivered to a honeypot, which is an isolated and monitored host that is used as a trap to collect further information about malicious activities. This redirection is done in a stealthy way without having the originator of the traffic observe that. This is very useful for detecting botnets, which are a set of connected compromised hosts controlled by an attacker and used as a platform to launch distributed attacks against other parts of the network

Therefore, the proposed logically centralized architecture supports highly reactive security monitoring, analysis and response systems to facilitate network forensics, security policy alteration and security service insertion. For network forensics, the proposed IDS/IPS facilitates quick and adaptive threat identification through a cycle of harvesting intelligence from the network to analyse network security, update security policies and reprogram the network accordingly.

4.5.4 Integration of IDS/IPS with SDN MANETs

The proposed hybrid IDS/IPS architecture naturally fits the four proposed SDN MANETs design options. Our MANETs design options use SDN controller replication for enhancing resilience in option 1 (D1) and option 2 (D2). Options 3 and 4 (D3 and D4) by design are distributed in nature with multiple SDN controllers to achieve both resilient, scalability

and load balance. In all the cases, the SDN controllers can be diversified by using different network operating systems and implementations from different vendors. We have also proposed in Chapter 3 to enhance the trust management for more secure control data communication, therefore enabling the IDS/IPS architecture to be more resilient to various forms of attacks, such as man-in-the-middle attacks, DoS attacks, and so on. Other mechanisms such as sampling, distributed local IDS/IPS processing, communication channel migration, multipath routing, self-healing, and dynamic secure re-association of switches to different SDN controllers can be implemented in the MANET architecture design options as well,

4.6 Summary

In this chapter, we reviewed the IDS/IPS system security capabilities and challenges in providing IDS in SDN MANETs. Various types of security threats and potential attacks have been analyzed from different perspectives. Three possible architecture designs for SDN MANETs were described We then proposed a hybrid IDS/IPS architecture and discussed in detail the component mechanism to provide scalable, flexible, resilient, and responsive protection against security threats to SDN MANETs. Finally, we examined the integration of the proposed IDS/IPS into our SDN MANET architecture design options.

Chapter 5

Simulation Development, Customization and Implementation

In this chapter we present the simulation environment and the implementation of the proposed SDN MANET architecture design options.

5.1 Introduction

There are very limited software choices for conducting simulation/emulation based studies of SDN networks.

Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet networks run real code including standard Unix/Linux network applications as well as the real Linux kernel and network stack. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking. As a result, Mininet creates a realistic virtual network, running real kernel, switch and application code, on a single machine. Nearly every operating system virtualizes computing resources using a process abstraction. Mininet uses process-based virtualization to run many hosts and switches on a single OS kernel. Mininet can create kernel or user-space OpenFlow switches, controllers to con-

trol the switches, and hosts to communicate over the simulated network. Mininet connects switches and hosts using virtual Ethernet pairs. Mininet currently depends on the Linux kernel. Mininet-based networks cannot exceed the CPU or bandwidth available on a single server. Unfortunately, Mininet cannot support the emulation of wireless links.

ns-3 is a discrete-event computer network simulator, primarily used in research and teaching. ns-3 is free software, publicly available under the GNU GPLv2 license for research, development, and use. ns-3 lacks the support for protocols (like WSN, MANET etc.) which were supported in ns-2. ns-3 is also not backward compatible with ns-2. ns-3 has OpenFlow support which for the time being is restricted to being simulation only. The design of the classes specific to providing SDN simulation capabilities in ns-3 primarily center on implementing an SDN controller and OpenFlow-enabled switch as user-defined applications. As with ns-2, ns-3 is also time consuming to learn and use compared to GUI-based simulators.

Other known network simulation software, such as QualNet, NetSim, GloMoSim, OMNeT++, do not appear to support SDN.

We are able to identify one network simulation software, EstiNet [92] that after customization can meet our needs. We will simulate various design options using EstiNet [92] 9.0 OpenFlow network simulator and emulator for our studies and evaluation.

EstiNet supports two modes of operations: the simulation mode and the emulation mode. In the simulation mode, a real-world open source OpenFlow controller such as NOX, POX, Floodlight, OpenDaylight and Ryu controllers can directly run on a controller node in the simulated network to control simulated OpenFlow switches without any modification. In the emulation mode, controller application programs can run on an external

machine that is different from the machine used to simulate OpenFlow switches to control these simulated OpenFlow switches. In [43] EstiNet is compared with Mininet and ns-3 regarding their capabilities, performance, and scalability. The research found that EstiNet has the characteristics of a simulator and an emulator at the same time. It combines the advantages of the simulation and emulation approaches without their respective shortcomings. EstiNet uses real OpenFlow controller programs, real network application programs, and the real TCP/IP protocol stack in the Linux kernel to generate correct, accurate, and repeatable SDN application performance results. Moreover, EstiNet support node mobility.

5.2 EstiNet Simulation Capabilities

A brief introduction of EstiNet's simulation capabilities are in order. EstiNet

- Uses the real-life Linux TCP/IP protocol stack to generate high-fidelity simulation results.
- Can run any real-life UNIX-based application program on a simulated node without any modification. Any real-life program (e.g., the Apache web server, P2P BitTorrent program, Skype VoIP program, VLC video streaming server and client program, DDoS (BoNeSi), or a program implemented in Java) can be run on a simulated host, router, mobile node, etc. to generate realistic network traffic.
- Can use any real-life UNIX network monitoring tools. For example, UNIX tcpdump, traceroute, and wireshark application programs can be run on a simulated network to monitor the simulated network.

- Setup and usage of a simulated network and application programs are exactly the same as those used in real-life IP networks.

- Simulates many types of important networks, for example, OpenFlow v1.3 enabled networks, IEEE 802.11n networks, IEEE 802.11(p)/1609 WAVE wireless vehicular networks, HLA (High-Level-Architecture) distributed emulations, IEEE 802.3 Ethernet-based fixed Internet, IEEE 802.11(a/g) wireless LANs, mobile ad hoc networks, IEEE 802.11(e) QoS wireless LANs, wireless vehicular networks (VANET) for Intelligent Transportation Systems (including V2V and V2I), etc.

- Simulates many important protocols, for example, Openflow, IEEE 802.11n, IEEE 802.3 CSMA/CD MAC, IEEE 802.11 (a/g) CSMA/CA MAC, IEEE 802.11(e) QoS MAC, wireless mesh network routing protocol, learning bridge protocol, spanning tree protocol, IP, Mobile IP, VoIP, DiffServ (QoS), RIP, OSPF, UDP, the four real-life TCP protocol versions used in the Linux kernel, RTP/RTCP/SDP, HTTP, FTP, Telnet, BitTorrent, POP3, SMTP, and all application-layer protocols used in real-life network applications.

- Executes and finishes a reasonably large network simulation case quickly.

- Generates reliable and repeatable simulation results.

- Provides a highly-integrated and professional GUI environment.

- Adopts a module-based architecture.

- Can be easily used as an emulator.

- Supports seamless integration of emulation and simulation.

While EstiNet provides a lot of features for researchers to conduction network simulation and emulation studies as shown above, to adapt EstiNet for supporting the simulation studies of our proposed SDN MANET architecture design options, a lof of software customization is necessary. The reasons are given as follows. All existing SDN controllers without any modification are not suitable for mobile ad hoc networks. Because all existing controllers are designed for controlling switch devices that have one layer-3 interface for the control plane and multiple layer-2 interfaces for the data plane, they are not suitable for being directly used to control mobile devices which, in this case, have one layer-3 interface for the control plane and one layer-3 interface (for layer-3 routing) for the data plane. And the OpenFlow protocol, which is used for the communication between a controller and a switch, is also not suitable for being directly used between a controller and a mobile node device. Additionally, EstiNet does not directly support the operations of our proposed heterogeneous networking environments and different network architecture variants. In the past year, we have been working the EstiNet development team to customize the EstiNet simulator to incorporate many additional modules and to implement new mechanism for supporting our simulation needs. More details of our customization of the simulation software and the simulation environments are described in the following sections.

5.3 SDN MANET Architecture Design Options in EstiNet

5.3.1 MANET Architecture Design Option 1

In this design option (shown in Figure 5.1), the control plane and data plane are logically separated. That means these two planes use the same wireless channel/medium to transmit

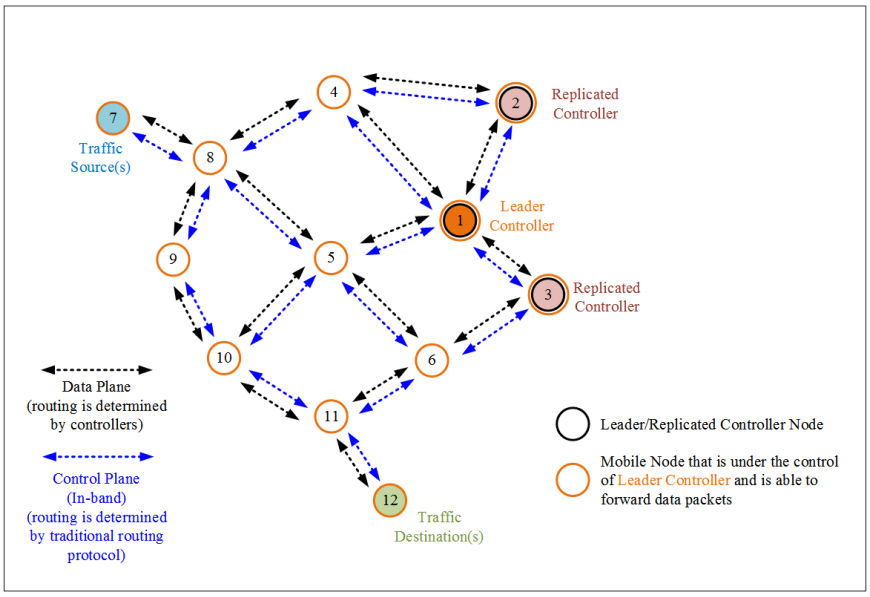

Figure 5.1: MANET Architecture Design Option D1.

messages but the routing decisions are made using different methods in each plane. In the control plane, the routing decision is made by a traditional routing protocol (AODV). In the data plane, the routing decision is made by the leader controller based on some criteria. The control plane and data plane can be separated using different IP subnet, for example, 1.0.1.x is for the data plane while 1.0.2.x is for the control plane. Figure 5.2 shows the implementation schematic diagrams of control nodes' and mobile nodes' protocol stacks used within the EstiNet simulation engine. In short, there is a routing mechanism, including a routing algorithm and a routing table, dedicated for the control plane while another routing mechanism, including only at the minimum a routing table, dedicated for the data plane. The routing algorithm for the data plane is built on an algorithm run at the SDN controller. The SDN controller configures a data plane routing table through the SDN controller and the OpenFlow protocol.

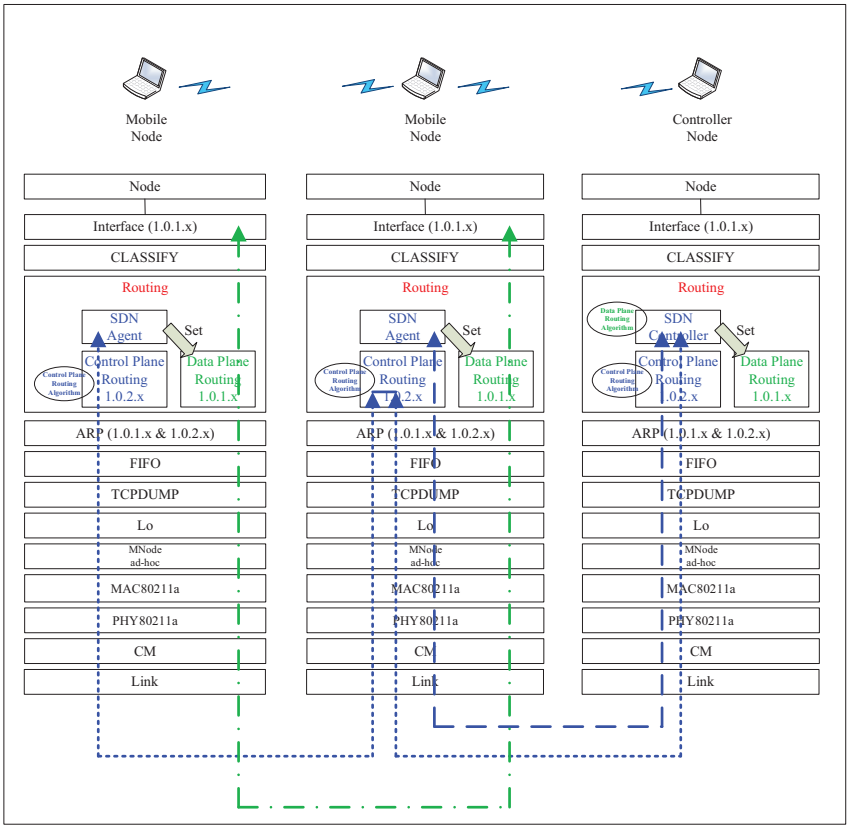

Figure 5.2: Simulation Engine for Design Option D1.

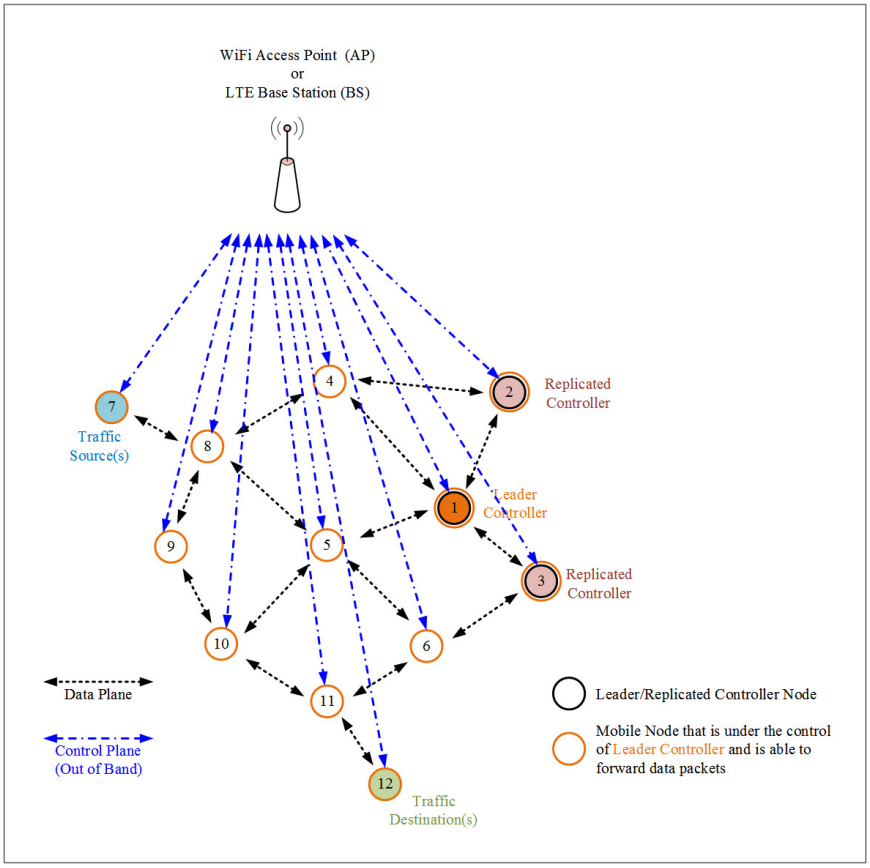

Figure 5.3: MANET Architecture Design Option D2.

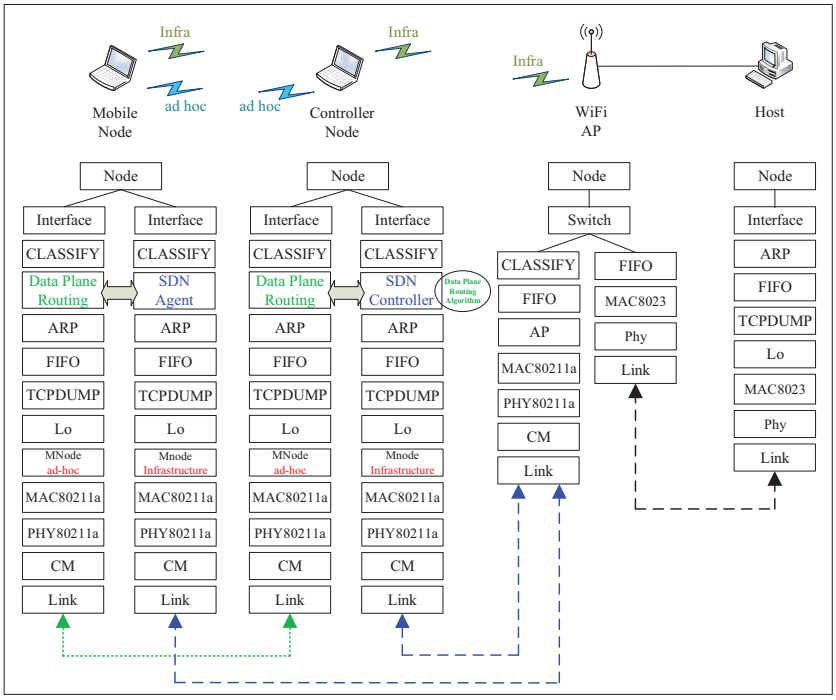

Figure 5.4: Simulation Engine for Design Option D2.

5.3.2 MANET Architecture Design Option 2

In this design option (shown in Figure 5.3), a separate infrastructure network (e.g., a WiFi LAN or a 4G network) is employed for the control plane. If we compare design option 2 (D2) with design option 1 (D1), we find that the major difference is that there is a separate infrastructure network for the control plane. In other words, this architecture uses an out-of-band communication for implementing the control plane. Figure 5.4 shows the implementation schematic diagrams of the control nodes' and mobile nodes' protocol stacks used within the EstiNet simulation engine.

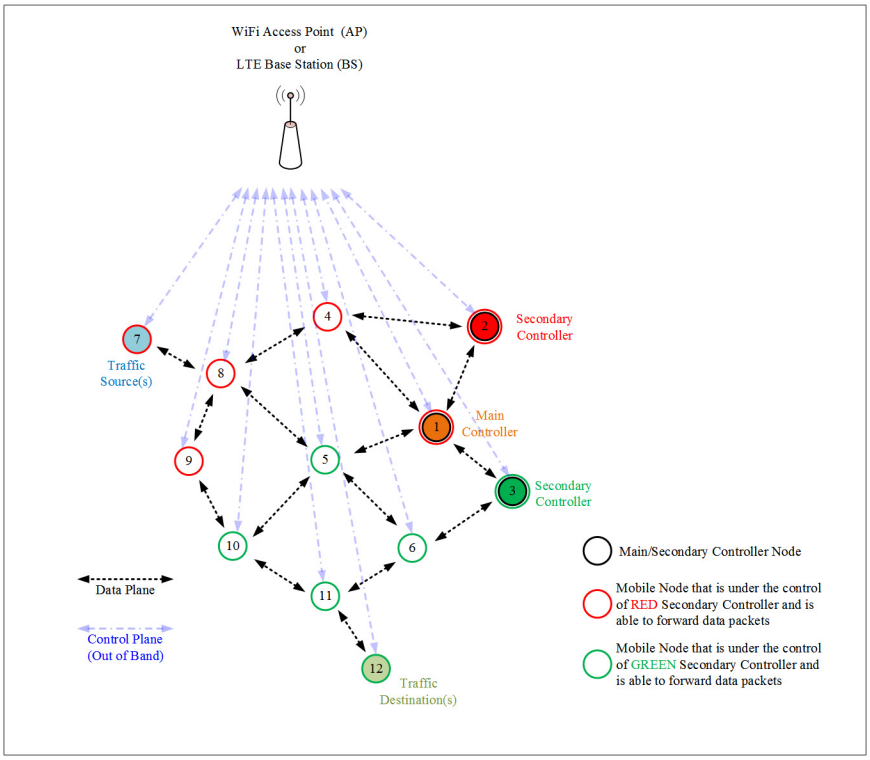

Figure 5.5: MANET Architecture Design Option D3.

5.3.3 MANET Architecture Design Option 3

In this design option (shown in Figure 5.5), the main controller delegates the routing function to secondary controllers. When a node is powered on and joins the MANET, it connects to the main controller and starts periodically updates of its neighbor list to the controller. The main controller learns and maintains the global view of the network. According to pre-determined network management policies (e.g., management load balancing among SDN controllers), the main controller may assign a newly arriving node to some secondary controller. When a node is assigned to a secondary controller, the node breaks the connection to the main controller first and then connects to the secondary controller. It then starts to update its neighbor list to the secondary controller. The updated neighbor list has to be sent to the main controller through the secondary controller so that the main controller still has the updated global view of the whole network. An assigned node could be re-assigned to another secondary controller based on the node's mobility and the pre-determined network management policies. The policies for assigning nodes to secondary controllers could be for load balancing, cluster forming and maintenance, and so on. In this work, we will implement only the policy for load balancing. That means all nodes that join the MANET will be assigned to secondary controllers so that the number of nodes managed by different secondary controllers will be as balanced as possible. Because a secondary controller does not have a global view of the entire network, it only configures its managed nodes with intra-cluster routing rules (i.e., routing within the set of nodes managed by the same SDN controller). When a node asks for inter-cluster routing rules (e.g., when a routing table lookup miss occurs), the node has to ask the secondary controller first for routing beyond

the current cluster. The secondary controller then asks the main controller for the required routing rules. The resulting routing rules are configured on the node by the secondary controller with limited lifetime (as defined by the network configuration). When a node is disconnected from its own cluster, it could be reassigned to another cluster or it could connect to its own cluster through other clusters. In the latter case, when the disjoint node wants to communicate with some nodes belonging to its original cluster, inter-cluster routing information is required which can then be obtained via the same procedure described above.

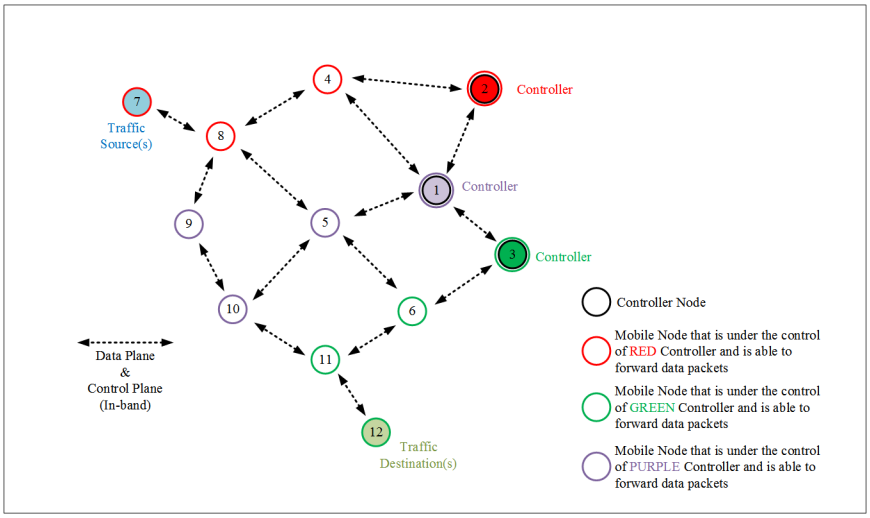

Figure 5.6: MANET Architecture Design Option D4.

5.3.4 MANET Architecture Design Option 4

The MANET SDN architecture employs full distributed controllers (as shown in Figure 5.6). The major design concept is that each controller manages its own nodes and shares its neighbor node list and each nodes neighbor list with other SDN controllers. Each node connects to its only controller and periodically updates its neighbor list to that controller. Over time, every controller will learn and maintain a global view of the network. As a

result, a controller can handle both intra-cluster and inter-cluster routing. Note that if needed, only the intra-cluster routing rules will be proactively maintained and produced on each managed node. The inter-cluster routing information will be kept on a controller until its managed nodes issue routing queries for this information. When a node is disconnected from its original cluster, it has to connect to its own cluster through other clusters. In this case, when the disjoint node wants to communicate with some nodes belonging to its original cluster, the inter-cluster routing information is required and can be obtained on-demand.

5.4 EstiNet Customization and MANET Architecture Implementation

Having described the details of the proposed SDN MANET architecture design options in EstiNet, we now present the efforts that we undertook to customize EstiNet and to implement the four architecture designs in EstiNet.

5.4.1 Design Options 1 and 2

- Control Plane Routing Module:

  We adopted the existing AODV module with some modification to direct the downlink packets to the ARP-and-Uplink-Dispatching module instead of the original ARP module.

- Data Plane Routing Module:

  We implemented routing table operations, including inserting, deleting, and searching for packet forwarding, and accepting routing table setting command from the

SDN Agent module.

- SDN Agent Module:

  A TCP connection is built to the leader controller at the initial phase. Hello messages are periodically sent to the SDN agent module of nearby mobile nodes using UDP datagrams. When receiving a Hello message, checking the neighbor list to see if the neighboring relationship is changed. If changed, update the changes into the list. Every time when the neighbor list changes, a neighbor update message is sent to the leader controller through the TCP connection. Upon receiving the route change messages from the leader controller through the TCP connection, the routing tables set in the Data Plane Routing Module located on the same node. When the TCP connection to the leader controller is broken, attempts will be made to connect to a replicated controller and exchange messages with that controller until this new connection is broken again. When the connection is broken again, attempts will be made to connect to the leader controller if it is reachable, otherwise, another replicated controller will be tried. A packet log is implemented to include time and some required packet information. Additionally, packet-based statistics are implemented, such as the number of incoming packets per second are recorded for evaluating the control plane overhead.

- ARP and Uplink Dispatching Module: A new module is created and added with the following features. First, two interface IP addresses are mapped into one MAC address by modifying the original ARP module. Second, the uplink packets are dispatched based on the IP subnet.

- Drop All Module: A new module is created and added which drops all incoming packets and outgoing packets.

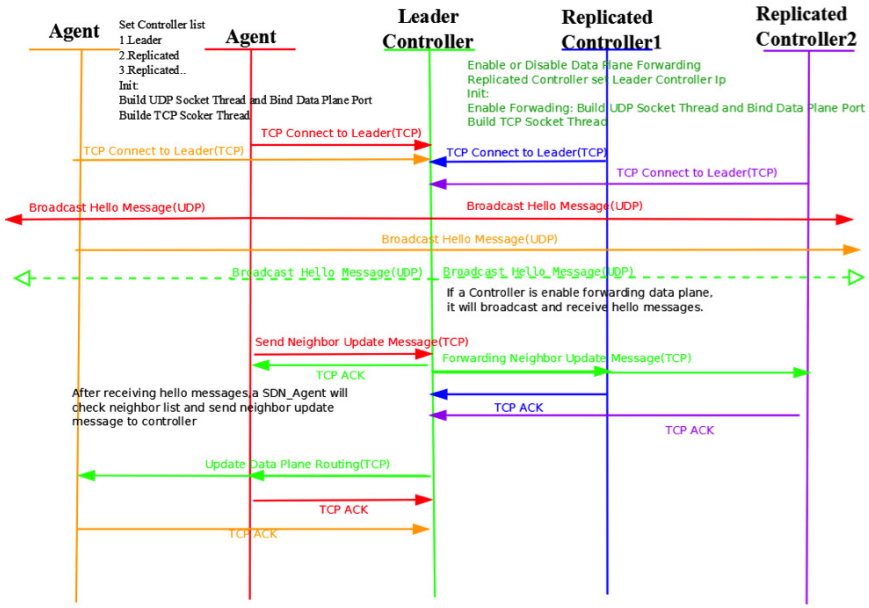

Figure 5.7: Design Option 1 (D1) and Design Option 2 (D2) Synchronization Mechanism between the Controllers.

The synchronization mechanisms between the controllers are depicted in Figure 5.7 and explained next.

The SDN controllers accept the TCP connection requests from SDN agent modules and maintain each connection in an SDN agent module list until the connection is broken.

Leader Controller The leader controller accepts the TCP connection requests from all replicated controllers for exchanging neighbor update messages sent from those SDN agent modules under their control. When the leader controller receives a neighbor update message from a mobile node (an SDN agent), it sends one copy to each other replicated controller and updates its network topology information. When the leader controller receives a neighbor update message from a replicated controller, it sends one copy to each other

replicated controller except the originating one and updates its own network topology information. When the network topology information is changed, the leader controller sends route change messages to those mobile nodes (SDN agents) that are under its control and have to update their routing tables.

Replicated Controller When a replicated controller receives a neighbor update message from a mobile node (SDN agent), it sends one copy to the leader controller and updates its network topology information. When a replicated controller receives a neighbor update message from the leader controller, it updates its network topology information. When the network topology information is changed, a replicated controller sends route change messages to those mobile nodes (SDN agents) that are under its control and have to update their routing tables. A shortest path first algorithm (e.g., the Dijkstra's algorithm) is implemented to determine the routing paths for the data plane based on the global network topology information collected. Additionally, a packet log is implemented to record time and some required packet information and packet-based statistics, such as number of incoming packets per second for evaluating the control plane overhead.

Based on the customization work for D1 and D2, additional software customization have been added to make the EstiNet simulator to support Options 3 and 4. These new additions included components for load-balancing mechanism network management.

5.4.2 Design Option 3

The synchronization mechanisms between the controllers are depicted in figure 5.8 and explained next.

- SDN Agent module:

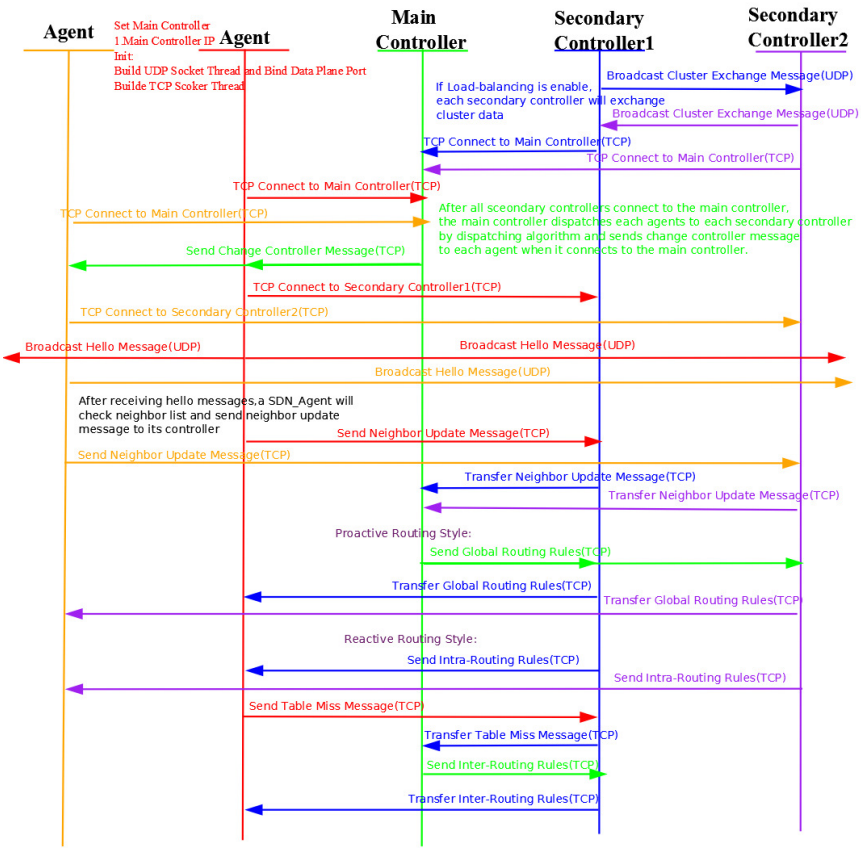

Figure 5.8: Design Option 3 (D3) Synchronization Mechanism between the Controllers.

Will create a TCP/IP connection to main SDN controller for run-time assigning to secondary SDN controller. It will create an UDP port for periodically broadcasting hello messages and receiving a hello message from other SDN agent or SDN controller modules. When receiving a change controller message, connect to a secondary SDN controller module. When receiving a Hello message, it will check its own neighbor list and send a neighbor update message to secondary SDN controller module. When receiving a routing change message, set the routing table in the data-plane routing module located on the same node. When the routing rule is missing, send a table miss message to secondary SDN controller module.

- SDN Controller module:

  It could be main or secondary controller:

  Secondary Controller, will create a TCP/IP connection to main controller for sending neighbor update messages or transferring table Miss messages that come from SDN Agent modules. It will transfer routing inter or global change messages that come from the main controller to those SDN Agent modules under its control. It will send routing intra change messages to those SDN Agent modules under its control if the routing protocol is reactive style. If the load-balancing is enable, broadcast cluster member information message to another secondary controller and execute the load-balancing algorithm periodically.

  Main Controller, It will dispatch an SDN Agent to a secondary controller by dispatching algorithm when it connects to the main controller. When receiving a neighbor update message, update its own topology and send routing global change mes-

sages to those secondary controllers if the topology is changed if the routing protocol is proactive-style. When receiving a table miss message, send routing inter change messages to the required secondary controller. A secondary controller does not have the whole network view, it only configures its managed nodes with intra-cluster routing rules. When a node asks for inter-cluster routing rules (a table miss occurs), the node has to ask the secondary controller first. The secondary controller then asks the main controller for the required routing rules.

Load-balancing mechanism for design option 3 includes four phases described in Figure 5.9 and explained below using an example (Figure 5.9).

- Initial Phase: In the initial phase, the secondary controller 2 controls the RED cluster that includes nodes 1, 2, 4, 7, 8, and 9. The secondary controller 3 controls the GREEN cluster that includes nodes 3, 5, 6, 10, 11, and 12.

- Changing Phase: In the changing phase, node 9 moves and disconnects from the RED cluster. However, it still has connectivity with node 10. The secondary controller 2 knows that node 9 leaves its cluster in the data plane. In turn, the main controller 1 knows that from secondary controller 2.

- Adaptive Phase I: When the orphan node joins another cluster, the secondary controller 2 asks the main controller 1 to designate another cluster for node 9 to join. In this case, the main controller 1 will tell secondary controller 2 to instruct node 9 to join secondary controller 3's cluster. For load-balancing purpose, the main controller 1 will also tell secondary controller 3 to choose one node within its cluster

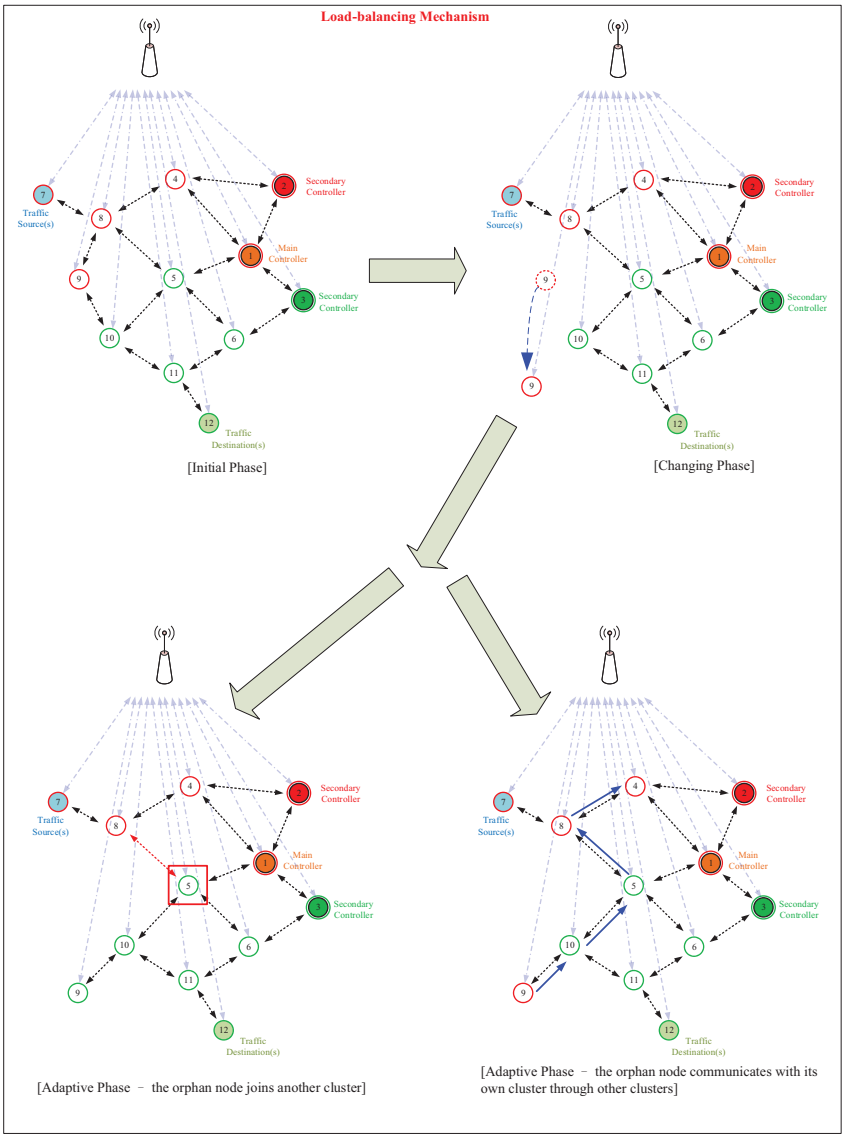

Figure 5.9: Load-balancing Mechanism for Design Option 3 (D3).

to join secondary controller 2's cluster. Secondary controller 3 will choose the node which has the most link connections to secondary controller 2's cluster and tell that node to change its cluster association. In this case the chosen node should be node 5. After the target node is chosen, secondary controller 3 will tell secondary controller 2 through main controller 1. If secondary controller 3 is not able to find one node within its cluster which has at least one connection with the secondary controller 2's cluster, it will inform main controller 1 of this situation and no node is re-assigned to secondary controller 2's cluster.

- Adaptive Phase II: When the orphan node communicates with its own cluster through other clusters, for example, if node 9 wants to send a message to node 4, there is a routing table lookup miss on node 4. Node 9 queries controller 2 for routing rules to go to node 4. Controller 2 asks controller 1 for routing rules because this requires inter-cluster routing. Controller 1 then answers "to go to node 4, the next hop is node 10" to node 9 through controller 2. The message arrives at node 10. Node 10 then asks controller 3 for routing rules to go to node 4. Controller 3 will then ask controller 1 for routing rules because this requires inter-cluster routing. Controller 1 answers "to go to node 4, the next hop is node 5" to node 10 through controller 3. The message arrives at node 5. Node 5 asks controller 3 for routing rules to go to node 4. Controller 3 in turn asks controller 1 for routing rules because this requires inter-cluster routing. Controller 1 answers "to go to node 4, the next hop is node 8" to node 5 through controller 3. The message arrives at node 8. Node 8 has a local cache of the routing rules to node 4 (i.e., intra-cluster routing). The message arrives

at node 4.

5.4.3 Design Option 4

The synchronization mechanisms between the controllers are depicted in figure 5.11 and explained next.

- SDN Agent module: The SDN agent module creates a TCP/IP connection to the SDN controller module, and an UDP port is created for periodically broadcasting Hello messages and receiving Hello messages from other SDN agents or SDN controller modules. When receiving a Hello message, it will check his own neighbor list and send a neighbor update message to the SDN controller module. When receiving a change controller message, it will connect to another SDN controller module. When receiving a routing change message, it will set the routing table in the data plane routing module located on the same node. When the routing rule is missing, a table miss message is sent to the SDN controller module.

- SDN Controller module: The SDN controller module creates a TCP/IP connection to the controller which has the least node ID for transferring neighbor update messages to another controller. Also it will send routing information change messages to those SDN agent modules under its control if the routing protocol is of the reactive-style. When receiving a table miss message, a routing change message is sent to the required SDN agent module. If the load-balancing is enabled, broadcast cluster member information message to another controller and execute the load-balancing algorithm periodically. When receiving a neighbor update message, update its own topol-

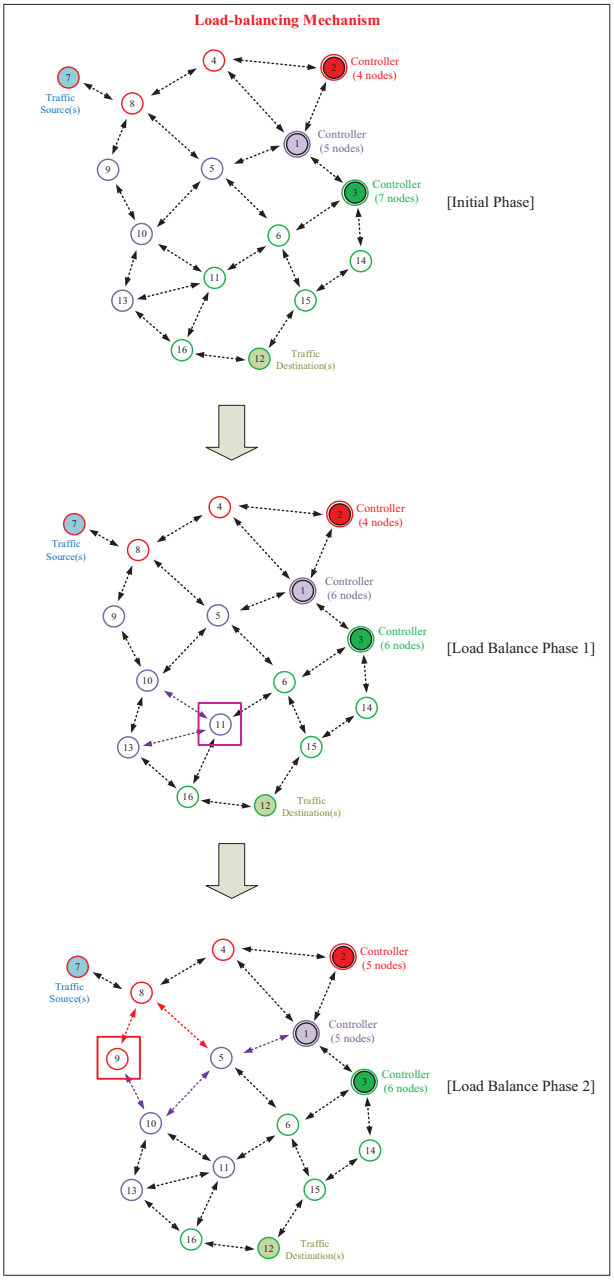

Figure 5.10: Load-balancing Mechanism for Design Option 4 (D4).

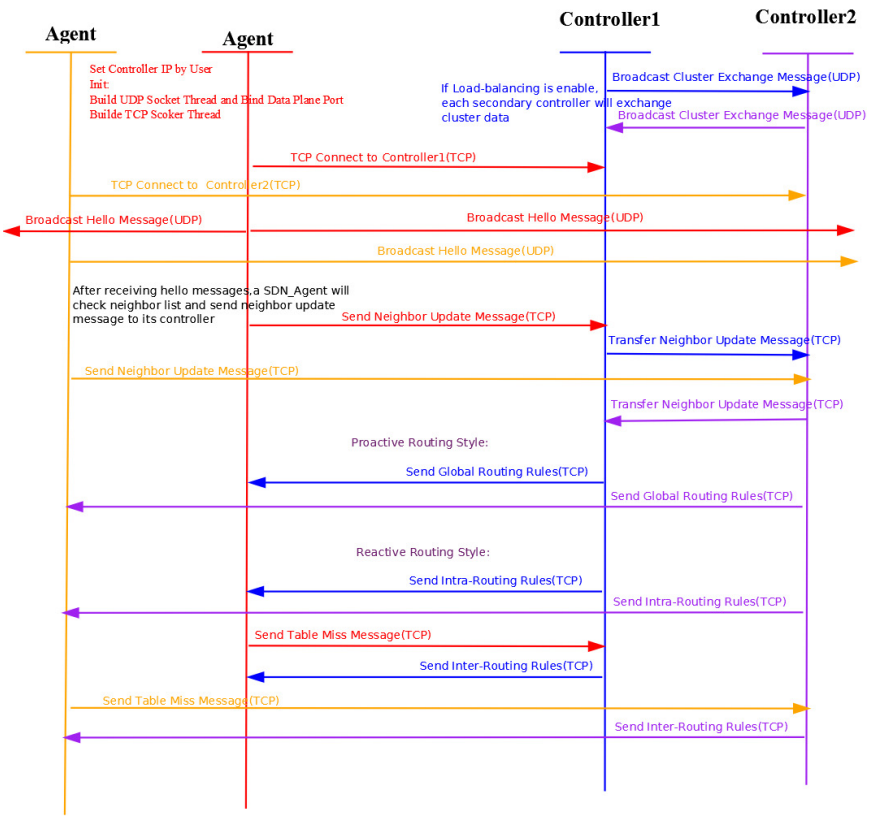

Figure 5.11: Design Option 4 (D4) Synchronization Mechanism between the Controllers.

ogy and send routing global change messages to those SDN agent modules under its control if the topology is changed and if the routing protocol is of the proactive-style.

Figure 5.10 shows the three phases of the load-balancing mechanism for design option 4. The details of the three phases are discussed below using an example.

- Initial Phase: Every node connects to its corresponding controller based on the network pre-configuration. Based on an all-controller list available locally, each controller communicates with other controllers to exchange cluster information, including the ID of each node and the neighbors of each node within a cluster. With this information, each controller can maintain a neighbor cluster table and a cluster routing table.

- Load balance Phase I: Periodically, each controller tries to balance the number of nodes within its cluster with its neighbor controller(s). For example, controller 1 has 5 nodes and has two neighbor clusters that are controlled by controllers 2 and 3. So controller 1 decides that there is no need to give away any node to them. Controller 2 has 4 nodes and has one neighbor cluster that is controlled by controller 1. Controller 2 decides that there is no need to give away any node. Controller 3's has 7 nodes and has one neighbor cluster that is controlled by controller 1. For load balance, controller 3 can give away (7-5)/2=1 node to its neighboring cluster. Controller 3 chooses the node which has the most connections to controller 1's cluster. In this case, the chosen node should be node 11 because it has 2 connections to controller 1's cluster, one is to node 10 and the other is to node 11. Controller 3 then tells node 11 to change its controller to controller 1.

- Load balance Phase II: Periodically, each controller tries to balance the number of nodes within its cluster with its neighbor controller(s). For example, controller 1 has 6 nodes and has two neighbor clusters that are controlled by controllers 2 and 3. So controller 1 decides that there is no need to give away any node to controller 3. It can give away (6-4)/2=1 node to controller 2. Controller 2 has 4 nodes and has one neighbor cluster that is controlled by controller 1. Controller 2 decides that there is no need to give away any node. Controller 3 has 6 nodes and has one neighbor cluster that is controlled by controller 1. Therefore, there is no need to give away any node. Controller 1 chooses the node which has the most connections to controller 1's cluster. In this case, the chosen node should be node 9 or node 5 because each of them has 1 connection to controller 2's cluster (not consider the controller node itself). In this case, controller 1 chooses the one with the least number of intra-cluster connections (for node 9 is 1, for node 5 is 2). Thus, node 9 is chosen. Controller 1 then tells node 9 to change its controller to controller 2. Now, controller 1 has 5 nodes, controller 2 has 5 nodes, and controller 3 has 6 nodes. The system reaches a balanced state.

5.5 Summary

In this chapter, we have summarized the simulation/emulation tools that are available for conducting simulation studies for SDN MANETs. We identified the strengths and weaknesses of various popular network simulation tools and determined that only EstiNet simulation software has the potential to support our studies. We briefly illustrated the capabilities of EstiNet simulator. We then presented the details of development, customization, and

implementation of our proposed SDN MANET architecture design options in the EstiNet 9.0 network simulator/emulator.

Chapter 6

Simulation Environment, Results and Discussion

In this chapter, we present the simulation setup, results and discussion.

6.1 Simulation Settings

6.1.1 Hardware and Software Specifications

We ran VMware workstation 12 pro version 12.0.0 on top of Windows Server 2008 R2 Enterprise Edition, 64-bit 6.1.7601, Service Pack 1. With Fedora 64-bit memory 3GB, dual core processor and 20 GB hard disk. Figure 6.1 and Figure 6.2 show the simulation environment for design option 1 and design option 2. Table 6.1 shows the simulation parameters.

6.1.2 Simulation Environments

Our simulation considered the four proposed architecture designs options detailed in the previous chapters. For each design we considered several simulation scenarios and collected a variety of performance metrics for evaluation. The four design option simulations are discussed below:

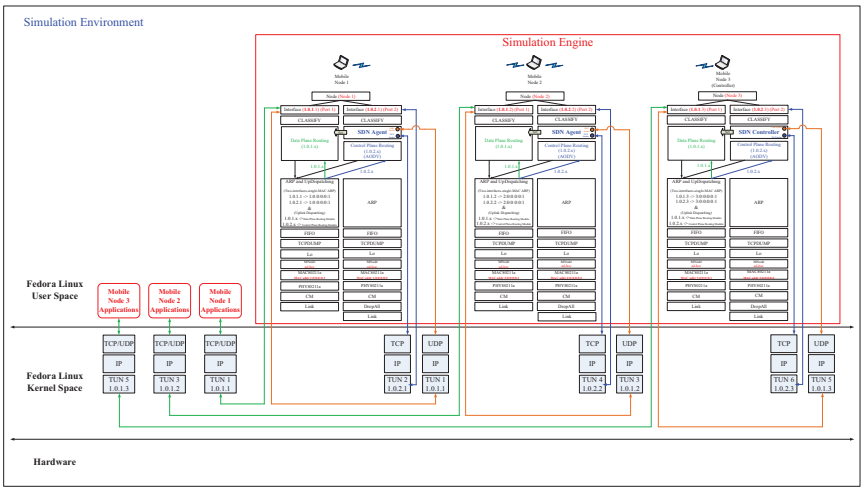

Figure 6.1: Design Option 1 (D1) Simulation Environment.

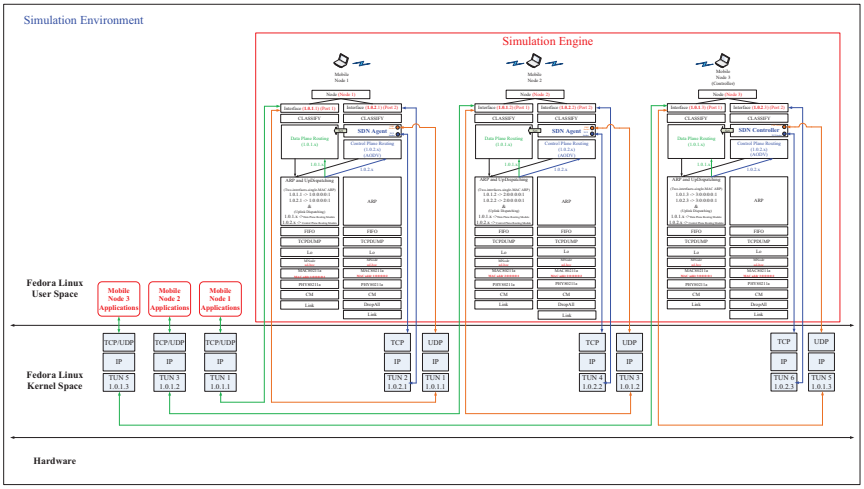

Figure 6.2: Design Option 2 (D2) Simulation Environment.

- Design option 1: Fully centralized SDN controller with controller replication. We ran the simulation for 5 topology scenarios and we studied the operations of the network architecture with respect to a number of metrics, such as control plane overhead, network throughput, packet loss, and power consumption.

  Figure 6.3 shows an example of our simulation network topology.

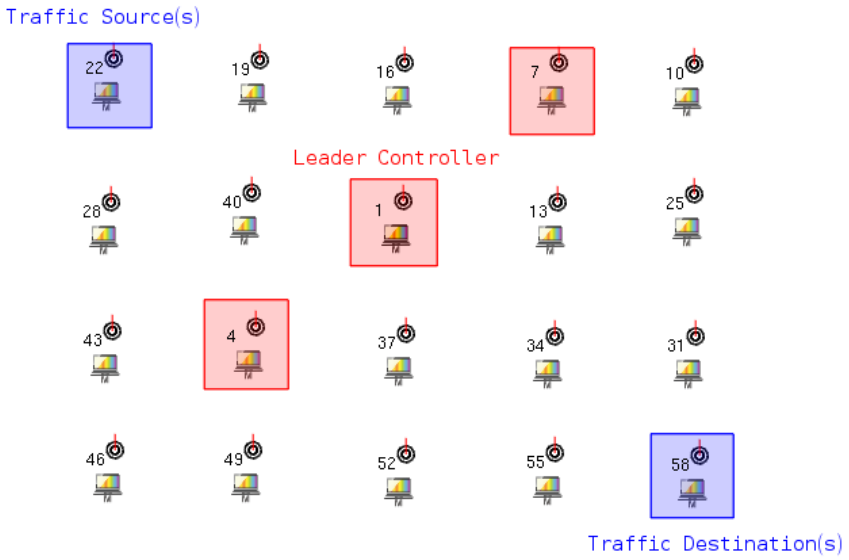

Figure 6.3: Design Option 1 (D1) Example Simulation Topology.

  - Simulation Time: 100 seconds
  - Leader Controller: Node 1
  - Replicated Controller: Node 4, Node 7
  - Traffic Source: Node 19
  - Traffic Destination: Node 34
  - Mobile Node Numbers: 20
  - Data Plane Traffic Rate: 1.35 Kbyte/sec

- Design option 2: Fully centralized SDN controller with controller replication and external infrastructure WiFi support for the control plane communication. We ran the simulation for 5 topology scenarios and we studied the operations of the network architecture with respect to a number of metrics, such as control plane overhead, network throughput, packet loss, and power consumption.

  Figure 6.4 shows an example of our simulation network topology:

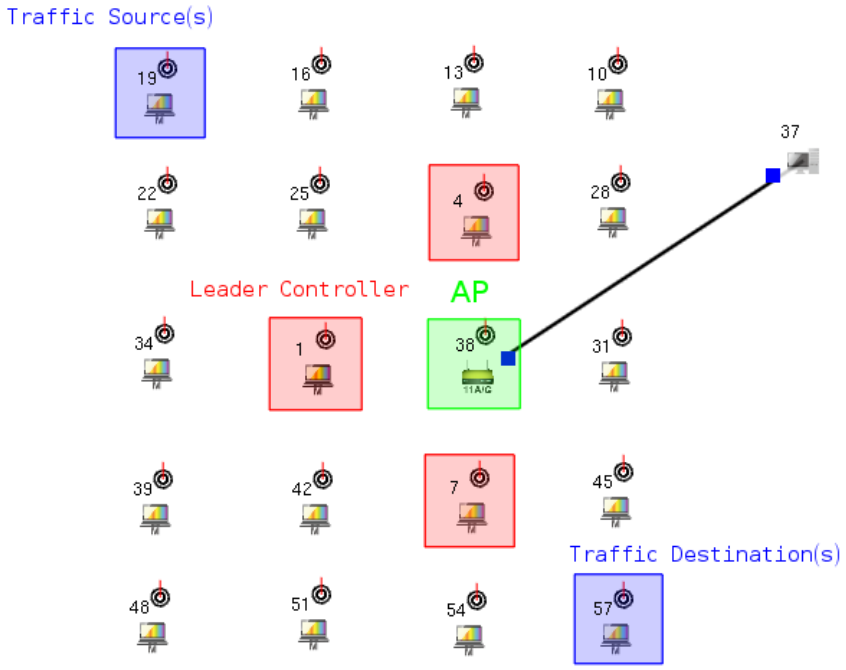

Figure 6.4: Design Option 2 (D2) Example Simulation Topology.

  - Simulation Time: 100 seconds
  - Leader Controller: Node 1
  - Replicated Controller: Node 4, Node 7
  - Traffic Source: Node 19
  - Traffic Destination: Node 34

- Mobile Node Numbers: 20
- Data Plane Traffic Rate: 1.35 Kbyte/sec
- WiFi Data Rate: 10 Mbps

Parameter	Value
Topology	10m x 30m
Network size	12, 20, 29
Number of Controllers	3
Channel	36
Frequency	5180 MHz
Data Rate	6.5 Mbps
Packet Size	51 Kbytes
Control Plane Mode	802.11 A/G
Data Plane Mode	802.11 n
Mobility Model	Random Waypoint
Moving Speed	10.0 m/sec
WiFi Data Rate	10 Mbps
Transmission Power	16.02 dbm
Receiving Sensitivity	-82.0 dbm
Transmission Time	5s-100s
Simulation Time	100s
Fail-over Period	50s-100s

Table 6.1: The Simulation Parameters.

- Design option 3: Hierarchical physically distributed SDN controllers with a logically centralized controller. In this design we ran the simulation for 5 different network sizes using reactive routing rules with and without load balancing. We tested our mechanisms in a pure ad hoc environment and in a heterogeneous environment using WiFi.

  Figure 6.5 and Figure 6.6 show examples to illustrate the network communication in a pure ad hoc environment and a heterogeneous environment with WiFi.

  - Simulation Time: 100 seconds
  - Leader Controller: Node 1

– Secondary Controllers: Node 4, Node 7

– Traffic Source: Node 16

– Traffic Destination: Node 10

– Data Plane Traffic Rate: 1.35 Kbyte/sec

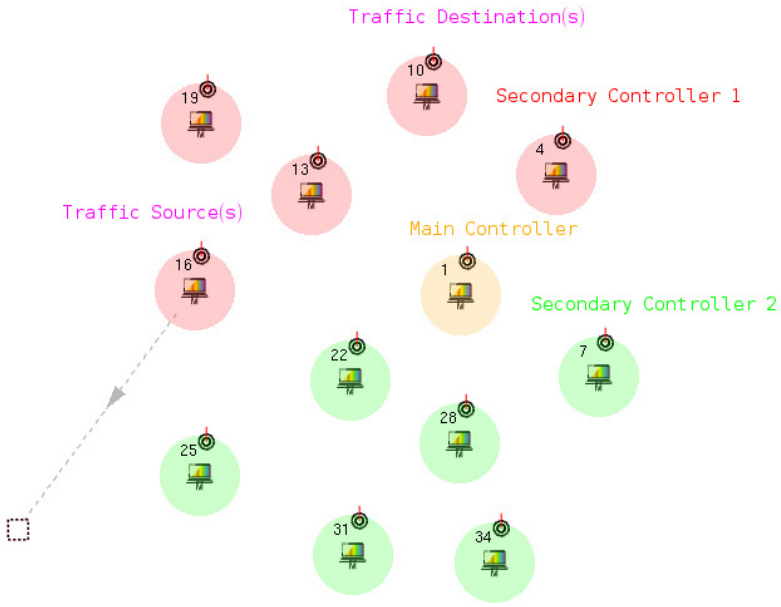

Figure 6.5: Design Option 3 (D3) A Pure Ad-hoc Environment.

In this example, the main controller will automatically dispatch node 10, node 13, node 16 and node 19 to join the secondary controller 1 and node 22, node 25, node 28, node 31 and node 34 to join the secondary controller 2. If a node receives a packet that cannot match any routing rules, the node will send a flowtable miss message to its controller (via reactive routing). When a mobile node moves from its original cluster to another cluster, if a load balancing mechanism is applied, the controller will instruct the mobile node to re-join another cluster. The controller will also attempt to balance the number of nodes in the clusters periodically. If a load

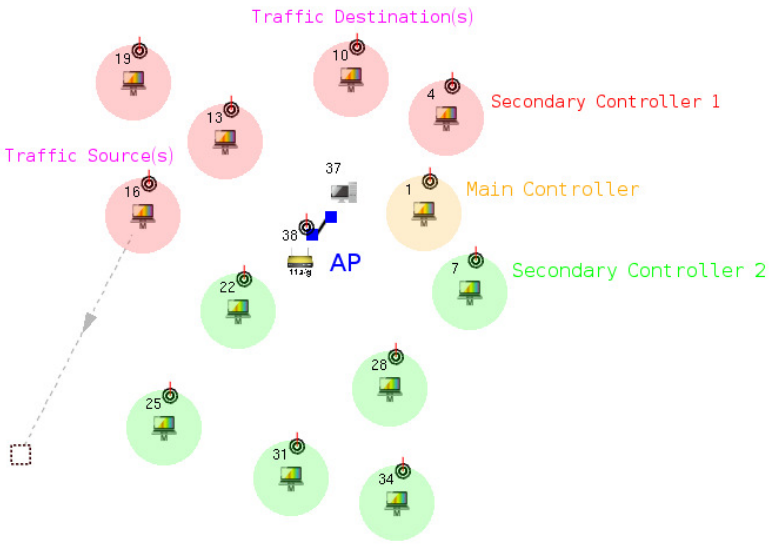

Figure 6.6: Design Option 3 (D3) A Heterogeneous Environment Using WiFi for Control Plane Communications.

balancing mechanism is not applied, if a mobile node moves from its original cluster to another cluster, the controller will not instruct the node to re-join another cluster. The controller will not attempt to balance the number of nodes in the clusters.

- Design option 4: Fully distributed but logically centralized SDN controllers. Similar to design option 3, we ran the simulation for 5 different network sizes using reactive routing rules with and without load balancing. We tested our mechanisms in a pure ad hoc environment and in a heterogeneous environment with WiFi. The difference between design option 3 and design option 4 is controllers in design option 3 are separated into a main controller and secondary controllers.

  Figure 6.8 shows an example to illustrate the network communication in a heterogeneous environment. There are three controllers, node 1, node 4, and node 7. All of them are multi-interface mobile nodes. In the control plane, they are all set to be with

a SDN-Controller module. Node 10, node 13, node 16, node 19, node 22 and node 25 are all multi-interface mobile nodes. In the control plane, they are all set to be with a SDN-agent module. Node 10 and node 13 are members of cluster 1 controller (node1). Node 16 and node 19 are members of cluster 2 controller (node 4). Node 22 and node 25 are members of cluster 3 controller (node 7). In the data plane, node 13 is set to be a traffic source and node 25 is set to be a traffic destination. Node 28 is an 802.11(a/g) Access Point and node 29 is a host.

Figure6.7 shows as an example to illustrate the network communication in a pure ad hoc environment. There are three cluster controllers, node 1, node 4 and node 7. All of them are multi-interface mobile nodes. In the control plane, they are all set to be with a SDN-Controller module. Node 10, node 13, node 16, node 19, node 22 and node 25 are all multi-interface mobile nodes. In the control plane, they are all set to be with a SDN-Agent module. Node 10 and node 13 are members of cluster 1 controller (node 1). Node 16 and node 19 are members of cluster 2 controller (node 4). Node 22 and node 25 are members of cluster 3 controller (node 7). In the data plane, node 13 is set to be a traffic source and node 25 is set to be a traffic destination.

If a node receives a packet that cannot match any routing rules, the node will send a flowtable miss message to its controller (via reactive routing). As in design option 3, if a load balancing mechanism is applied, when a mobile node moves from its original cluster to another cluster, the controller will instruct it to re-join another cluster. The controller will also attempt to balance the number of nodes in the clusters periodically. If no load balancing mechanism is applied, when a mobile node moves

from its original cluster to another cluster, the controller will not instruct it to re-join another cluster. The controller will not attempt to balance the number of nodes in the clusters.

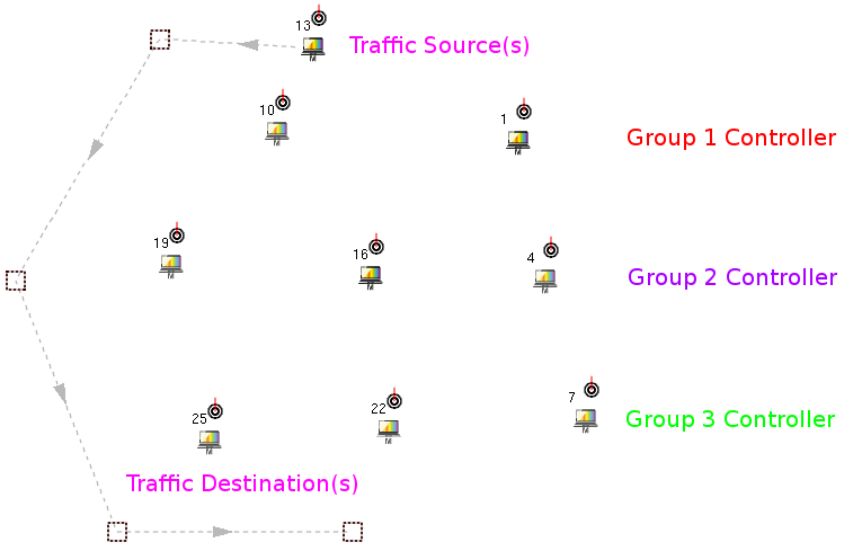

Figure 6.7: Design option 4 (D4) Ad-hoc Environment Control Plane Communications.

6.2 Simulation Results

Design Option 1: Fully centralized controller with controller replication

The simulation was performed for network size of 10, 20, 30, 40 and 50 nodes using EstiNet 9.0 network simulator in an area of size 10 m x 30 m. With three controllers, one acts as a leader controller and two as replicated controllers. The performance metrics such as total throughput, packet loss rate, power consumption and controller overhead are evaluated with and without the leader controller fail-over. The purple color curves represent the non fail-over scenario while the green color curves represent the fail-over scenario.

Figure 6.9 shows that the largest total throughput is achieved when the network size is 10 nodes and it is slightly more in fail-over scenario than in non fail-over scenario. With

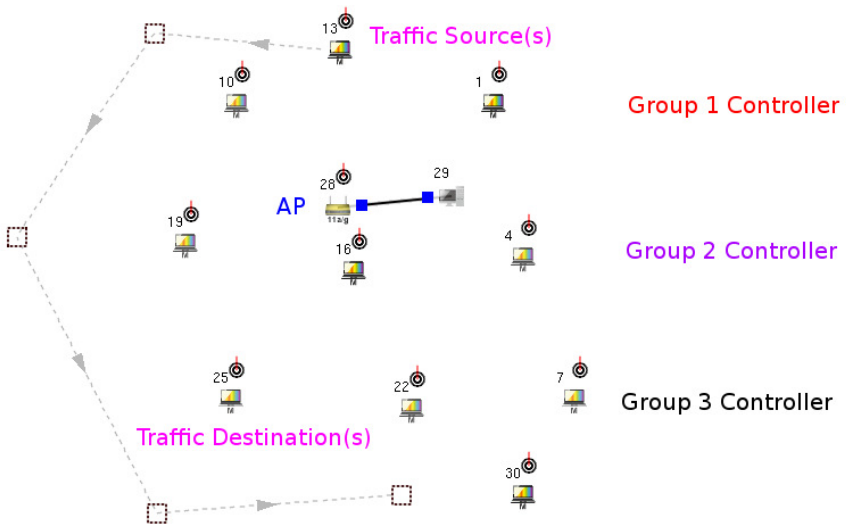

Figure 6.8: Design Option 4 (D4) Heterogeneous Environment using WiFi for Control Plane Communications.

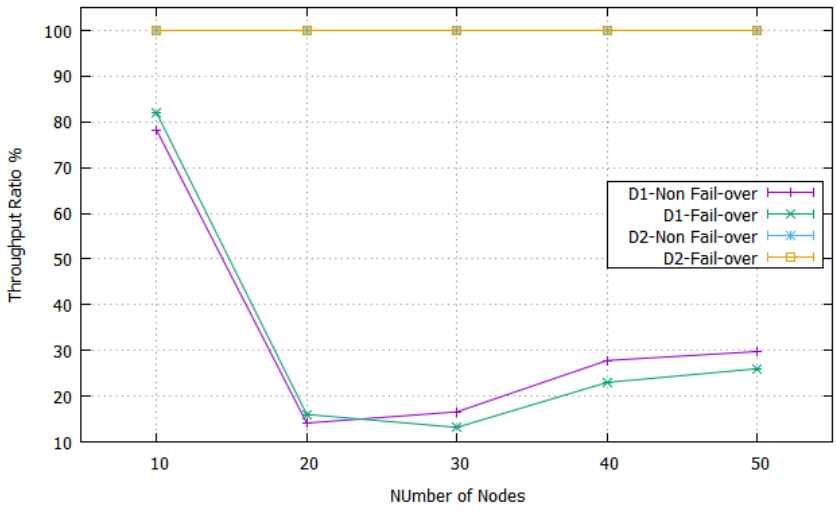

Figure 6.9: Design Option 1 (D1) and Design Option 2 (D2) Network Throughput for Fail-over and Non Fail-over Scenarios.

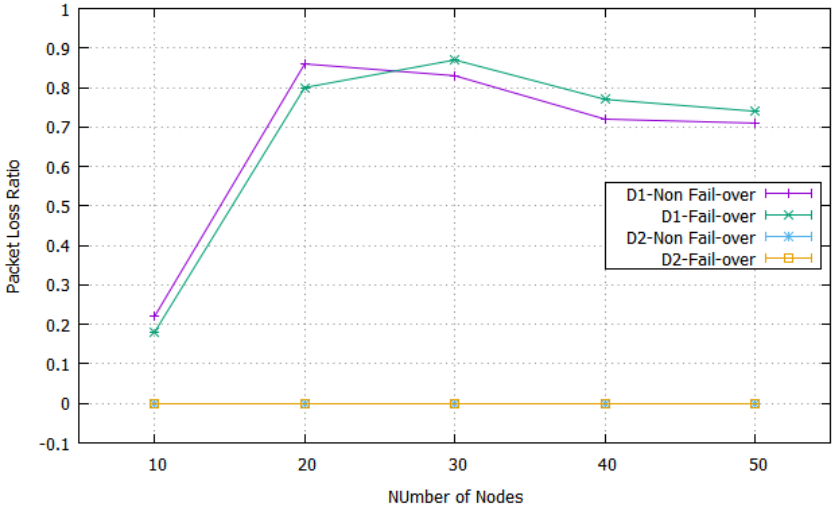

Figure 6.10: Design Option 1 (D1) and Design Option 2 (D2) Packet Loss Rate for Fail-over and Non Fail-over Scenarios.

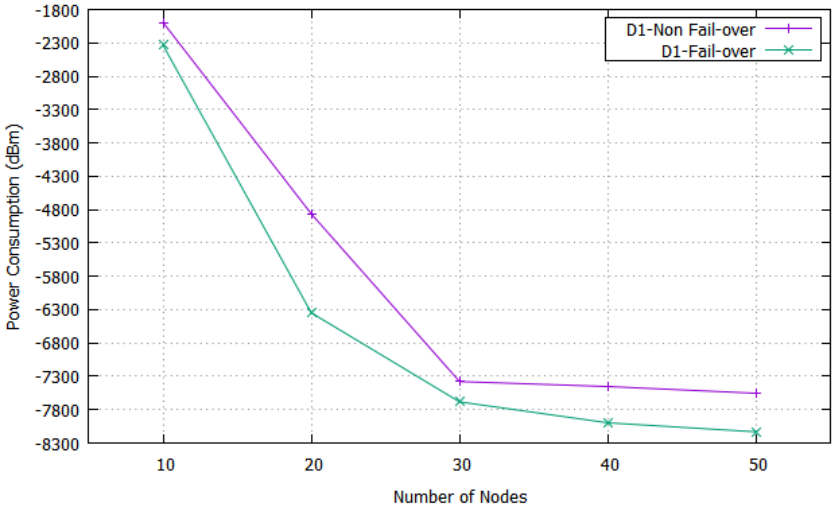

Figure 6.11: Design Option 1 (D1) Average Power Consumption for Fail-over and Non Fail-over Scenarios.

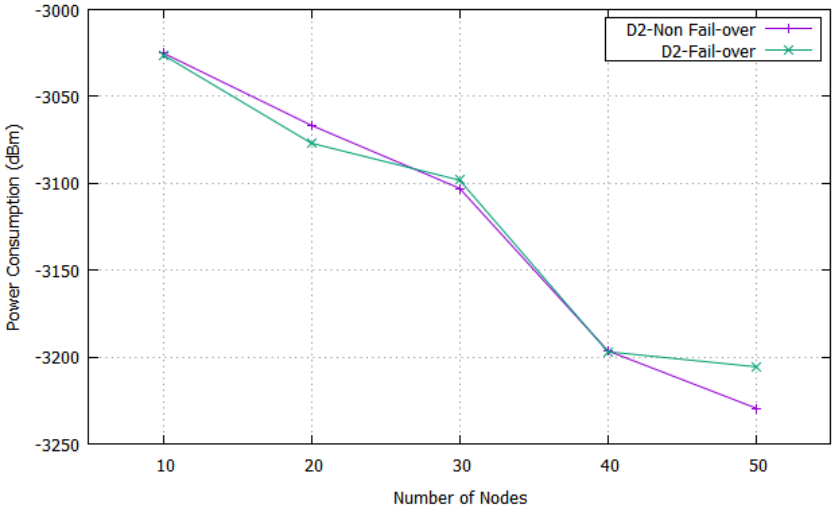

Figure 6.12: Design Option 2 (D2) Power Consumption for Fail-over and Non Fail-over Scenarios.

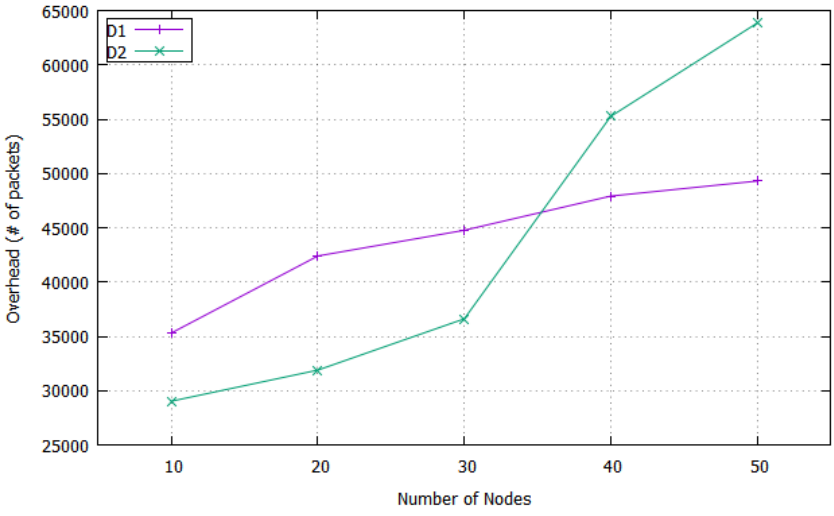

Figure 6.13: Design Option 1 (D1) and Design Option 2 (D2) Total Overhead.

increasing network size the total throughput decreases for both scenarios.

Figure 6.10 shows that the packet loss rate increases with increasing network size for both non-fail-over scenarios and fail-over scenarios. However, the packet loss rate is higher in the non fail-over scenarios when the network size is small (10, 20 nodes), and the packet loss is higher in the fail-over scenario than in the non-fail-over scenario with larger network size.

The network power consumption is also evaluated in our experiments. We measured the TX power and the RX sensitivity. The TX power is the total output power of transmitters (Transmission Energy) and the RX sensitivity is the receiver sensitivity (Reception Energy). We obtained the total power consumption for two scenarios, fail-over and non-fail-over for network size of 10, 20, 30, 40 and 50 nodes. We found that the maximum power consumed when the network size is 10 nodes and gradually degrades with 20, 30, 40 and 50 nodes for both scenarios. However, in fail-over scenario the power consumption is slightly less than non-fail-over scenarios. Figure 6.11 shows the total power consumption.

As the throughput and the packet drop increase with increasing number of nodes (i.e., larger network size), we found the controller overhead increases as well (Table 6.2). Figure 6.13 shows the controller overhead (in terms of number of packets) versus the throughput for network size of 10, 20, 30, 40 and 50 nodes.

Design Option 2: Fully centralized SDN controller with controller replication and external infrastructure support

The simulation has been performed for network size of 10, 20, 30, 40 and 50 nodes using EstiNet 9.0 Network Simulator in an area of size 10 m x 30 m. With three controllers, one acts as a leader controller and two as replicated controllers. WiFi with a data rate

of 10 Mbps is used the external supporting infrastructure. Performance metrics such as throughput ratio, packet loss rate, power consumption and controller overhead are evaluated under two scenarios: with or without leader controller fail-over. The blue color curves represent the non-fail-over scenario while the yellow color curves represent the fail-over scenario.

Figure 6.9 shows that the total throughput ratio is higher in non-fail-over scenarios than in fail-over scenarios. When the network size increases, the controller fail-over does not seem to affect the network total throughout. Figure 6.10 shows that the packet loss rate is lower for non-fail-over scenarios than in fail-over scenarios. The increasing network size and the controller fail-over do not seem to significantly affect the packet loss rate.

Power consumption is also evaluated in our experiments. We measured the TX power and the RX sensitivity. We obtained the total power consumption for our two scenarios, fail-over and non fail-over with network size of 10, 20, 30, 40, and 50 nodes. We found that the maximum power consumed when the network size is 10 nodes and gradually degrades for both scenarios (Figure 6.12).

Finally, the controller overhead (number of packets) is evaluated. The overhead increases with increasing network size (Table 6.2).

Design Option 3: Hierarchical physically distributed controllers

The simulation has been done for network size of 10, 20, 30, 40 and 50 nodes using EstiNet 9.0 network simulator in an area of size 10 m x 30 m. With three controllers, one act as a main controller and two as a secondary controllers. The performance metrics such as network throughput ratio, control plane overhead and power consumption are evaluated under two scenarios, load balancing and no load balancing for pure ad hoc and heterogeneous

environments.

Figure 6.15 shows the network throughput for the ad hoc mode with load balancing and no load balancing. We found that the network throughput is higher when there is no load balancing mechanism applied with increasing network size. The network throughput ratio in a heterogeneous environment is very high for all cases.

Figure 6.14 shows the average power consumption. We can see that if load balancing is disabled, the power consumption is less than when the load balance is enabled for both ad hoc and heterogeneous environments, and in the ad hoc environment less power is consumed than in the heterogeneous environment.

We evaluated the controller overhead. The lowest overhead is found in the ad hoc environment when load balancing is enabled and the overhead increases with increasing network size (Figure 6.16). The controller overhead in the heterogeneous environment when the load balancing is enabled is lower than the controller overhead when the load balancing is disabled.

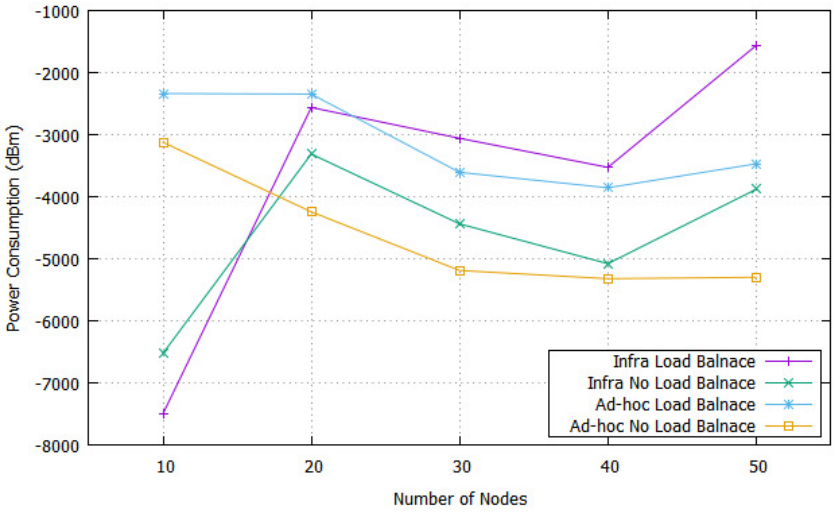

Figure 6.14: Design Option 3 (D3) Average Power Consumption for Ad-hoc Environment and Heterogeneous Environment with and without Load-balancing.

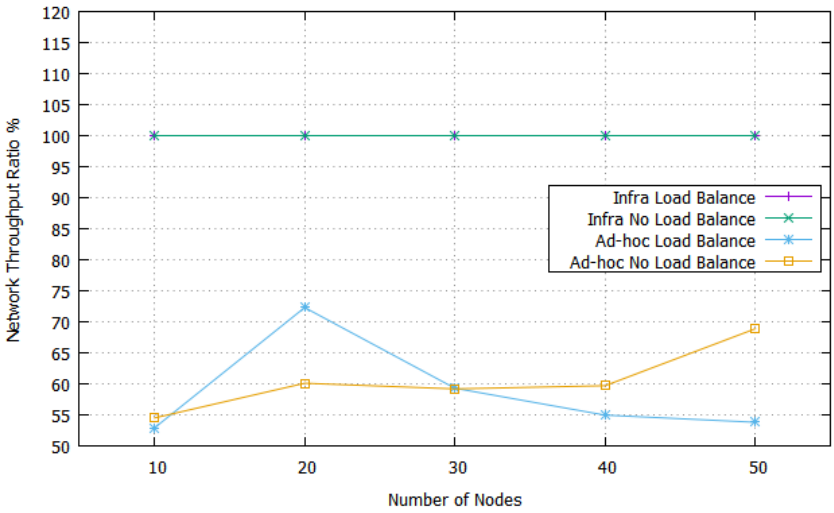

Figure 6.15: Design Option 3 (D3) Network Throughput for Ad-hoc and Heterogeneous Environments with and without Load-balancing.

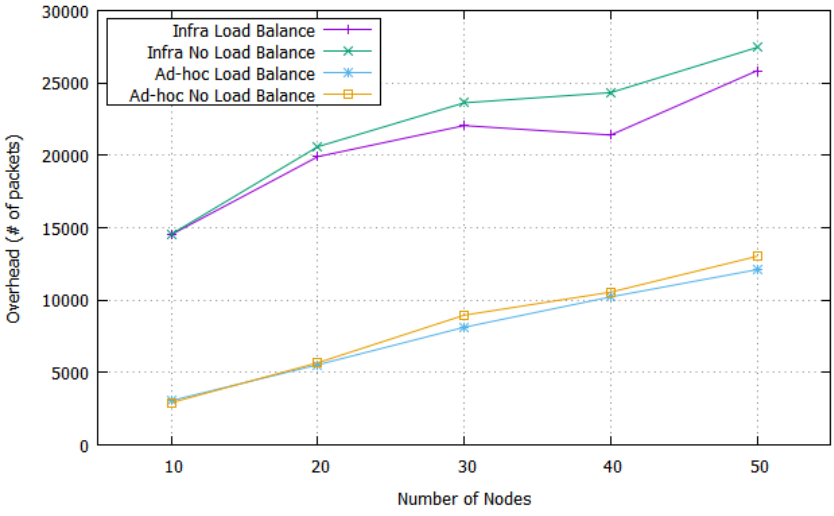

Figure 6.16: Design Option 3 (D3) Total Overhead.

Design Option 4: Fully distributed but logically centralized SDN controllers

The simulation has been performed for network size of 10, 20, 30, 40 and 50 nodes using EstiNet 9.0 network simulator in an area of size 10 m x 30 m. With three distributed controllers. The performance metrics such as network throughput ratio, total control plane overhead and power consumption are evaluated under two scenarios, load balancing and no load balancing for pure ad hoc and heterogeneous environments.

Figure 6.18 shows the network throughput ratio for ad-hoc environment and heterogeneous environment with and without load balancing. We found that the network throughput for ad-hoc environment is the highest with and without load balancing mechanisms when the network size is small (10 nodes). The throughput decreases with increasing network size (20 nodes) and stays almost stable for network sizes of 20 nodes, 30 nodes, 40 nodes and 50 nodes. And we found that the network throughput is very high for heterogeneous environment with and without load balancing.

Figure 6.17 shows the average power consumption. We can see that if load balancing is enabled the power consumption is slightly less than when the load balance disabled for both ad hoc environment and heterogeneous environment, and less power is consumed in ad hoc environment than in heterogeneous environment.

We also evaluated the control plane overhead. The control plane overhead is less if load balancing mechanism is applied. The lowest overhead is found in ad hoc environment when load balancing is enabled and increasing with increasing network size (Figure 6.19).

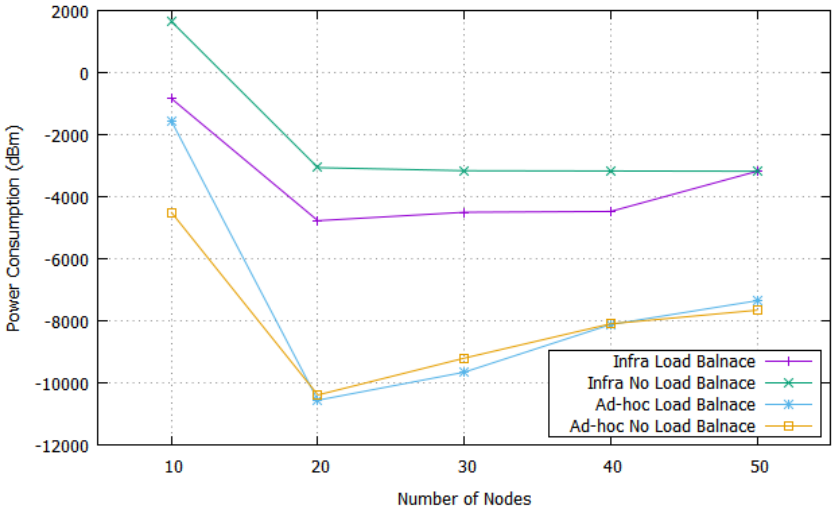

Figure 6.17: Design Option 4 (D4) Power Consumption in Ad-hoc Environment and Heterogeneous Environment with Load-balancing and No-load-balancing.

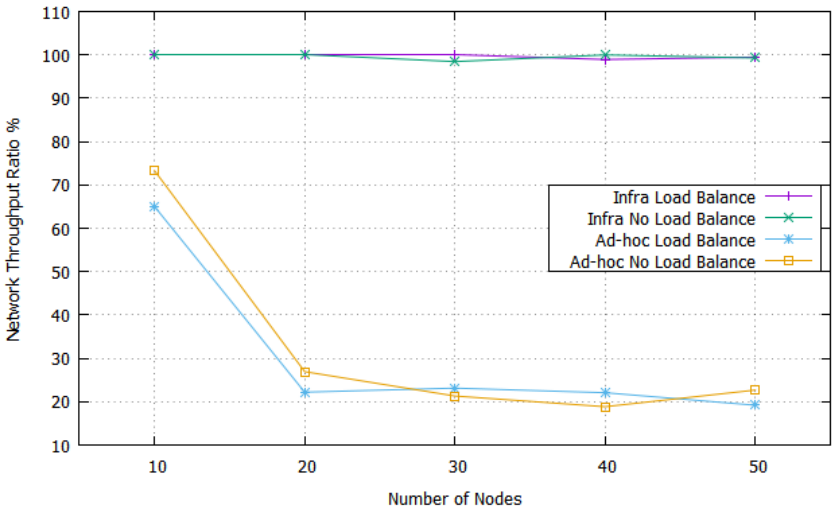

Figure 6.18: Design Option 4 (D4) Network Throughput for Ad-hoc and Heterogeneous Environment with Load Balancing and no Load Balancing.

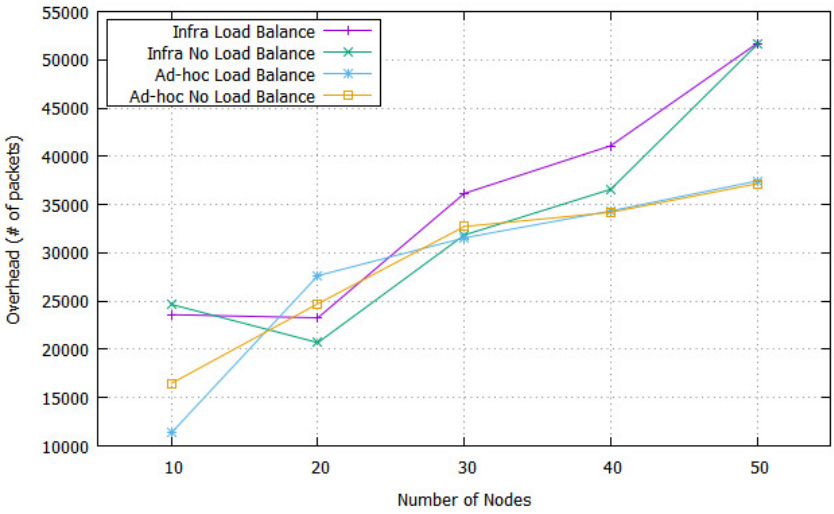

Figure 6.19: Design Option 4 (D4) Control Plane Overhead (number of packets) for Heterogeneous and Ad-hoc Environment with and without Load Balancing.

6.3 Discussion

6.3.1 Centralized SDN controller with Controller Replication (D1 and D2)

We evaluated our design options using various performance metrics.

Network Throughput is defined as the total delivered data packets (successfully send and receive packets) divided by the total transmitted packets during the simulation time (100s).

For design option 1, fully centralized controller with controller replication We found that throughput decreases with increasing network size due to increasing packet drop. The total throughput in fail-over scenarios is almost same as that of non fail-over scenarios. This indicates the design overcomes the fault tolerance problem since the replicated controller handled the leader controller fail-over.

For design option 2, fully centralized controller with controller replication and WiFi external infrastructure support. We found that the total throughput is always high for both scenarios due to low packet drop rate. In this design option, the control plane is out of band

using the external infrastructure (WiFi). That provides stability and reduces communication load for the nodes. The total throughput in the fail-over scenarios is almost the same as that of non-fail-over scenarios.

Figure 6.9 shows the throughput ratio versus the number of nodes for design option 1 and design option 2. From the figure we can see that the throughput ratio in design option 2 is better than design option 1 due to the external infrastructure support in design option 2. This external support provided stability and high throughput ratio for design option 2. For both designs replicated controllers perform very well, the throughput ratio is almost the same with both fail-over and non-fail-over scenarios. Our results show that the fault-tolerance problem could be mitigated with replicated controller architecture without sacrificing the performance.

Packet Loss Rate is defined as the total number of dropped packets divided by the total number of data packets by sources. For design option 1, fully centralized controller with controller replication. Our results show that the packet loss rate increases with increasing network size for both non-fail-over scenarios and fail-over scenarios due to increasing packet drop. Packets are dropped when a network node is carrying more data than it can handle. In this design option, the network carries two types of traffic data-plane traffic and control-plane traffic that may cause the network to be congested therefore increasing the packets that are dropped.

For design option 2, fully centralized controller with controller replication and WiFi external infrastructure support. Packet loss rate is always low for non-fail-over scenarios and in fail-over scenarios due to low packet drop. In this design option, the control plane traffic is carried by the external WiFi support network, and only the data plane traffic is

carried by the ad hoc network. That reduces the traffic on the data plane, resulting in lower packet drops.

Figure 6.10 shows the total packet loss rate versus the number of nodes for design option 1 and design option 2. From the figure we can see that the total packet loss rate in design option 2 is better than design option 1 due to the external infrastructure support in design option 2. This external support provided stability and low packet loss rate for design option 2. Both designs replicated controllers perform very well, and the total packet loss rate is almost the same in both fail-over and non fail-over scenarios. Our results show that the fault-tolerance challenge could be overcome with replicated controller architecture without sacrificing the performance.

Power Consumption Nodes in MANETs spend their energy in four cases. These cases are: transmission mode, reception mode, idle mode and overhearing mode. Nodes are in transmission mode when they send data packets to other nodes in the network. These nodes require energy to transmit data packets. Such energy is called Transmission Energy (Tx) [64]. When a node receives data packets from other nodes, the energy taken to receive packets is called Reception Energy (Rx) [64]. Idle mode is neither transmitting nor receiving any data packets. But this mode consumes power because the nodes have to listen to the wireless medium continuously in order to detect an arriving packet so that the node can then switch into receive mode from idle mode [86]. When a node receives data packets that are not destined for it, then it said to be in over-hearing mode [11]. In our experiments, we considered transmission mode and reception mode only.

For design option 1, we found that as shown in Figure 6.11 the total energy consumption for non-fail-over and fail-over scenarios dramatically degrades with increasing number of

nodes. The distance between nodes is different for different network sizes. The smaller network size may lead to longer distances between nodes than the distance between nodes in larger/dense networks. Therefore smaller networks need more power than larger/dense networks as smaller networks have longer transmission links and need more transmission power than larger networks.

For design option 2, we found that as shown in Figure 6.12 the total energy consumption for non-fail-over and fail-over scenarios slightly degrades with increasing number of nodes. Nodes in heterogeneous environment have a separate infrastructure network for the control plane. In other words, it is an out-of-band control plane. MANETs nodes consume most of their energy for forwarding data packets. In our scenario nodes send and receive just data plane messages (in-band). The energy consumed is only slightly affected by the network size.

Comparing design option 1 and design option 2 power consumption (Figure 6.11 and Figure 6.12), we found that design option 2 for large network size consumes more power than design option 1 due to traffic transmission in two bands (in-band for data plane traffic and out-band for control plane traffic) where in design option 1 data plane traffic and control plane traffic are all forwarded in-band.

For both designs fail-over scenario does not consume more power than non fail-over scenario.

Control Plane Overhead is defined as the total routing and control packets generated during the simulation time. For design option 1, fully centralized controller with controller replication, the controller overhead increases when the network size increases. For design option 2, fully centralized controller with controller replication and external WiFi infras-

tructure support. The controller overhead increases when the network size increases. Figure 6.13 and table 6.2 show the controller overhead (in terms of number of packets) versus the throughput for different network sizes of (10, 20, 30, 40 and 50) nodes.

For both designs controller overhead increases with increasing network size. Controller overhead is less in design option 2 with small network sizes (10 nodes, 20 nodes and 30) and more with increasing network size (40 nodes and 50 nodes).

The routing mechanism of the control plane for the infrastructure environment (D2) is different from the ad-hoc environment (D1). In the infrastructure environment (D2), routing is accomplished by the SDN via information collected through the infrastructure. However, in the ad-hoc environment (D1), routing uses AODV. AODV uses packet broadcast for route discovery. We found that the collisions in the ad-hoc environment (D1) is more than the infrastructure environment (D2). As a result, the SDN agent needs more time to connect to the SDN controller in option D1. When the SDN agent is temporarily disconnected from the SDN controller, it takes time for the SDN agent to reconnect to the SDN controller.

The actual communication time (after the SDN agent connected successfully to the SDN controller) in an ad hoc environment (D1) is less than in an infrastructure environment (D2). We obtained the overhead based on the packets between the SDN agent and the SDN controller when the SDN agent has connected successfully to the SDN controller.

In general if we compare design option 1 and design option 2, we will find design option 2 outperforms design option 1 by throughput, packet loss rate and communication reliability. However, the power consumption is higher than design option 1, but is not affected significantly with increasing network size.

6.3.2 Distributed SDN Controller Architecture (D3 and D4)

Network Throughput For design option 3, in ad hoc environment, the network throughput is higher when there is no load balancing mechanism with increasing network size. In the heterogeneous environment, network throughput is very high due to the WiFi support to the control plane (packet drop is approximately 0). We also found that network throughput is high when load balancing is not applied. When a mobile node moves from its original cluster to another cluster, the controller will instruct the mobile node to re-join another cluster. This process affects the network throughput and increases packet drop.

For design option 4, if we compare network throughput for ad-hoc environment and heterogeneous environment we found that network throughput for heterogeneous environment is very high and for ad hoc environment is very low with increasing network size due to increasing packet drop.

Power Consumption For design option 3, the average power consumption is less when the load balancing is disabled than when the load balance enabled for both ad hoc environment and heterogeneous environment due to the extra packet forwarding (table miss, inter-cluster routing and intra-cluster routing) when the load balancing is enabled. In addition, ad hoc scenario consumes less power than heterogeneous environment due to in-band traffic transmission (data plane traffic and control plane traffic both transmitted in-band).

For design option 4, the average power consumption is slightly less when load balancing is enabled than when the load balance is disabled for both ad hoc environment and heterogeneous environment because when the load balancing is disabled the probability of packet collision increases which requires retransmission. As the number of packet trans-

mitted increases, the power consumption increases.

In addition, the ad hoc scenario consumes less power than in the heterogeneous environment due to the in-band traffic transmission (i.e., the data plane traffic and the control plane traffic is both transmitted in-band).

Comparing power consumption in design option 3 and design option 4, design option 4 consumes less power than design option 3 due to on demand inter-cluster routing.

Control Plane Overhead For design option 3, the control plane overhead is less if load balancing mechanism is applied in both ad hoc and heterogeneous environments, and the overhead is more in heterogeneous environment (Table 6.3 and Table 6.4).

The load balancing mechanism reduces the controller overhead in both environments. When a load balancing mechanism is applied, the controller will instruct the mobile node to re-join another cluster when the mobile node moves from its original cluster to another cluster and the controller attempts to balance the number of nodes in the clusters periodically. When a nodes receives a packet that cannot match any routing entries in their flowtable, a flow table miss message is sent to the controller and the controller will send back the inter-cluster routing entry to the mobile node.

If no load balancing mechanism is applied, the controller will not instruct the node to re-join another cluster if a mobile node moves from its original cluster to another cluster. The controller will not attempt to balance the number of nodes in the cluster, resulting in lower overhead.

When load balancing is applied, the probability of packet collision for load balancing mechanism appears to be higher than when no load balancing is applied due to increasing traffic (inter-cluster routing and intra-cluster routing). As a result, it takes longer for the

SDN agent to connect to the SDN controller. Therefore, this reduces the opportunities for effective communication when load balancing is applied, leading to lower overhead.

The overhead when using load balancing in ad hoc environment is also lower compared to that in an infrastructure environment for the same reason that discussed above for design option 1 (D1). The AODV routing uses packet broadcast for route discovery. We found that the packet collisions in the ad-hoc mode is more than the infrastructure mode. And it takes more time for the SDN agent to get connected to the SDN controller in this case. When the SDN agent is temporarily disconnected from the SDN controller, the SDN agent has to reconnect to the SDN controller. It takes less time for the SDN agent to get connected successfully to the SDN controller in an ad hoc environment than in an infrastructure environment, leading to lower overhead in an ad hoc environment.

For design option 4, the control plane overhead is higher when load balancing mechanism is applied in both ad hoc and heterogeneous environments and it is more in at heterogeneous environment (Table 6.3 and Table 6.4).

When load balancing is applied, the overhead is higher in an ad hoc environment and in a heterogeneous environment due to increasing number of packets (send and receive). And the overhead is higher in a heterogeneous environment because of the actual communication time in a heterogeneous environment is more than that in an ad hoc environment.

In general design option 3 and design option 4 have high performance in an infrastructure environment. Design option 4 is more reliable than design option 3 because of lower probability of packet collision and disconnection from the controller. Design option 3 provides better throughput in an ad hoc environment.

Metrics	10 nodes	20 nodes	30 nodes	40 nodes	50 nodes
Design Option 1	35311	42403	44768	47915	49304
Design Option 2	29026	31890	36613	55251	63851

Table 6.2: Total Control Plane Overhead (number of packets) for Design Option 1 (D1) and Design Option 2 (D2).

6.4 Summary

In this chapter, we reported our simulation results and performance evaluation of the proposed SDN MANET architecture design options. The evaluation has been performed with respect to different network sizes, traffic scenarios, fail-over scenarios, and using various performance metrics, including throughput, packet drop, packet loss rate, power consumption, and so on. We provided extensive discussions of the simulation results and the pros and cons these different architecture design options.

Metrics	10 nodes	20 nodes	30 nodes	40 nodes	50 nodes
Design Option3-Load Balancing	3063	5535	8127	10224	12110
Design Option3-NO-Load Balancing	2920	5675	8960	10554	13025
Design Option4-Load Balancing	11365	27641	31554	34342	37452
Design Option4-No-Load Balancing	16475	24734	32729	34203	37166

Table 6.3: Total Control Plane Overhead (number of packets) for Design Option 3 (D3) and Design Option 4 (D4) in an Ad-hoc Environment.

Metrics	10 nodes	20 nodes	30 nodes	40 nodes	50 nodes
Design Option3-Load Balancing	14531	19911	22058	21405	25484
Design Option3-NO-Load Balancing	14566	20603	23630	24334	27454
Design Option4-Load Balancing	23595	23268	36161	41100	51701
Design Option4-No-Load Balancing	24658	20697	31846	36588	51582

Table 6.4: Total Control Plane Overhead (number of packets) for Design Option 3 (D3) and Design Option 4 (D4) in A Heterogeneous Environment.

Chapter 7

Conclusions and Future Work

We conclude this dissertation by summarizing our contributions and identifying several new directions for future research in the area of software defined secure mobile ad hoc networks.

7.1 Major Contributions

Security is a critical component of a mobile ad hoc wireless network. Achieving security and performance in a mobile ad hoc wireless network poses significant challenges. This dissertation focuses on software defined secure mobile ad hoc wireless networks by designing and evaluating various network architecture options, mechanisms for establishing initial trust among participating nodes, as well as intrusion detection mechanisms in SDN based MANETs. The proposed designs will offer better security and trust management by taking advantages of the software defined networking paradigm that enables more flexibility in programming and managing mobile ad hoc wireless networks.

The contributions of our work are three folds:

- We propose, study, and evaluate software defined network architecture for collaborative secure ad hoc wireless networking. Specifically, we presented four design

options of SDN MANET architecture for collaborative secure ad hoc wireless networks. These design options fall into two types: (1) centralized controller with controller replication and (2) distributed controller architecture.

In centralized controller with controller replication architecture, we proposed and studied two design options (a) fully centralized controller with controller replication; and (b) fully centralized controller with controller replication and external network infrastructure support, such as cellular 4G and/or WiFi networks. For the distributed controller architecture, we proposed two different schemes (c) hierarchical integrated physically distributed controllers to achieve a logically centralized controller; and (d) fully distributed but logically centralized controllers. In both scenarios, a control platform is implemented in multiple servers located in the network and collectively provides a programmatic interface (i.e., a control logic) to implement management features such as routing, access control, trust, and security.

While these proposed architecture options exhibit different characteristics, many common challenges are shared amongst these options. Challenges include fault-tolerance, scalability, efficiency, and security. The unstructured nature of ad hoc wireless networks exacerbates these challenges. We have studied the pros and cons of these different design options and their applicability in different practical scenarios via simulations.

- We addressed the most fundamental issue of providing initial trust between nodes in MANETs. To enable a secure SDN mobile wireless ad hoc network, in particular for setting up initial trust among the nodes in SDN MANETs, such as between the SDN

controllers and other network nodes, we have proposed the design of a lightweight virtual PKI. Rather than running a physical PKI within the SDN MANET or relying on the availability of a PKI that is accessible only via a supporting infrastructure, we proposed to provide the equivalent PKI capabilities and services by creating a PKI with trusted virtual certificate authorities (VCAs). We proposed and studied trusted virtual certificate authorities (VCAs) based local infrastructure for supporting device mutual authentication to support secure communications/operations in SDN based MANETs, and therefore, relieving the MANETs of the need to rely on an external public key infrastructure (PKI). We have examined the integration of the VCA infrastructure with the different design options of the SDN based MANET architecture.

- We reviewed the IDS/IPS system security capabilities and challenges in providing IDS in SDN MANETs. Various types of security threats and potential attacks have been analyzed from different perspectives. Three possible architecture designs for SDN MANETs were described. We then proposed a hybrid IDS/IPS architecture and discussed in detail the component mechanism to provide scalable, flexible, resilient, and responsive protection against security threats to SDN MANETs. Finally, we examined the integration of the proposed IDS/IPS into our SDN MANET architecture design options.

7.2 Future Directions

In this dissertation, while we have addressed some issues and challenges for enabling the secure and collaborative operations of SDN mobile ad hoc wireless networks, many research issues and challenges in the general SDN MANET area remain unsolved. More specifically related to this work, due to limited resources and time, a few tasks will be left as future work.

Trust Establishment We studied the initial trust establishment based on authentication using a virtual certificate authority for SDN based MANETs, as a way to deal with heterogeneous networking in terms of both technology heterogeneity and device heterogeneity. Authentication using a virtual certificate authority originates from the traditional public-key infrastructure based approach for establishing initial trust among parties. We provided some theoretical analysis of this virtual certificate authority based scheme and its integration with different MANET SDN architecture. Future work needs to implement the scheme and evaluate its effectiveness and performance via simulation and/or emulation for different SDN based secure MANET architecture in terms of the message overhead, energy consumption, and scalability. Moreover, using zero knowledge proof to establish initial trust among participants in SDN MANETs is another possible approach to explore. To the best of our knowledge, no study has been performed on utilizing zero knowledge proof to establish initial trust among participants in SDN networks.

Intrusion Detection and Prevention systems We described a theoretical design of a hybrid IDS/IPS architecture for SDN MANETs to provide a scalable, resilient, extensible, and responsive approach to enhancing the protection of such networks. The design incor-

porates numerous component mechanisms. These mechanisms need to be carefully orchestrated in an MANET SDN paradigm and implemented using high-level programming languages such as Pyretic and evaluated via emulation in order to quantify their effectiveness and overhead. A tradeoff may have to be made in order to strike a proper balance among possibly conflicting objectives and performance factors.

Experimental Evaluation While we have experimented and evaluated our proposed architecture design options with extensive simulation studies, additional experimental evaluation will be needed to test the applicability of the architecture options for more specific application scenarios and on a larger scale.

Bibliography

[1] L. Abusalah, A. Khokhar, and M. Guizani. A survey of secure mobile ad hoc routing protocols. *IEEE Communications Surveys & Tutorials*, 10(4):78–93, 2008.

[2] W. J. Adams and N. J. Davis IV. Toward a decentralized trust-based access control system for dynamic collaboration. In *Proceedings of the 6th Annual IEEE SMC on Information Assurance Workshop*, pages 317–324. IEEE, 2005.

[3] A. Adnane, C. Bidan, and R. de Sousa. Trust-based countermeasures for securing OLSR protocol. In *Proceedings of the IEEE on Computational Science and Engineering*, volume 2, pages 745–752. IEEE, 2009.

[4] E. Aivaloglou, S. Gritzalis, and C. Skianis. Trust establishment in ad hoc and sensor networks. In *Critical Information Infrastructures Security*, pages 179–194. Springer, 2006.

[5] P. Albers, O. Camp, J. Percher, B. Jouga, L. Mé, and R. S. Puttini. Security in ad hoc networks: a general intrusion detection architecture enhancing trust based approaches. In *Proceedings of Wireless Information Systems*, pages 1–12, 2002.

[6] M. A. Ayachi, C. Bidan, T. Abbes, and A. Bouhoula. Misbehavior detection using implicit trust relations in the AODV routing protocol. In *Proceedings of the IEEE on Computational Science and Engineering*, volume 2, pages 802–808. IEEE, 2009.

[7] V. Balakrishnan, V. Varadharajan, U. K. Tupakula, and P. Lues. Trust and recommendations in mobile ad hoc networks. In *Proceedings of the 3rd IEEE International Conference on Networking and Services*, pages 64–64. IEEE, 2007.

[8] J. Ballard, I. Rae, and A. Akella. Extensible and scalable network monitoring using opensafe. *Proc. INM/WREN*, 2010.

[9] J. S. Baras and T. Jiang. Cooperative games, phase transitions on graphs and distributed trust in MANET. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, volume 1, pages 93–98. IEEE, 2004.

[10] S. Bengio, G. Brassard, Y. G. Desmedt, C. Goutier, and J. Quisquater. Secure implementation of identification systems. *Journal of Cryptology*, 4(3):175–183, 1991.

[11] A. Bhardwaj, S.Sofat, et al. An efficient energy conserving scheme for ieee 802.11 adhoc networks. In *Wireless and Optical Communications Networks, 2007. WOCN'07. IFIP International Conference on*, pages 1–5. IEEE, 2007.

[12] F. Borran, R. Prakash, and A. Schiper. Extending paxos/Lastvoting with an adequate communication layer for wireless ad hoc networks. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems, SRDS'08*, pages 227–236. IEEE, 2008.

[13] A. Boukerche and Y. Ren. A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks. In *Proceedings*

of the 5th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, pages 88–95. ACM, 2008.

[14] S. Buchegger and J. Le Boudec. Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings of the IEEE Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403–410. IEEE, 2002.

[15] S. Buchegger and J. Le Boudec. Performance analysis of the CONFIDANT protocol. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 226–236. ACM, 2002.

[16] M. Burmester, P. Kotzanikolaou, and C. Douligeris. Security in mobile ad hoc networks. *Network Security: Current Status and Future Directions*, pages 355–374, 2006.

[17] A. T. Campbell, I. Katzela, K. Miki, and J. Vicente. Open Signaling for ATM, Internet and Mobile Networks (OPENSIG'98). *ACM SIGCOMM Computer Communication Review*, 29(1):97–108, 1999.

[18] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: taking control of the enterprise. *ACM SIGCOMM Computer Communication Review*, 37(4):1–12, 2007.

[19] B. Chang and S. Kuo. Markov chain trust model for trust-value analysis and key management in distributed multicast MANETs. *IEEE Transactions on Vehicular Technology*, 58(4):1846–1863, 2009.

[20] B. Charron-Bost and A. Schiper. The heard-of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, 2009.

[21] J. Cho, A. Swami, and R. Chen. Modeling and analysis of trust management for cognitive mission-driven group communication systems in mobile ad hoc networks. In *Proceedings of the IEEE International Conference on Computational Science and Engineering*, volume 2, pages 641–650. IEEE, 2009.

[22] J. Cho, A. Swami, and R. Chen. A survey on trust management for mobile ad hoc networks. *IEEE on Communications Survey & Tutorials*, 13(4):562–583, 2011.

[23] C. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang. Nice: Network intrusion detection and countermeasure selection in virtual network systems. *Dependable and Secure Computing, IEEE Transactions on*, 10(4):198–211, 2013.

[24] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo. Software defined wireless networks: unbridling SDNs. In *Proceedings of the European Workshop on Software Defined Networking (EWSDN)*, volume 1, pages 1–6, Oct 2012.

[25] C. R. Davis. A localized trust management scheme for ad hoc networks. In *Proceedings of 3rd International Conference on Networking*, pages 671–675, 2004.

[26] J. R. Douceur. The sybil attack. In *Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[27] A. Ejaz, S. Kashan, and M. Waqar. Cluster-based intrusion detection (CBID) architecture for mobile ad hoc networks. In *Proceedings of the 5th Asia Pacific Informa-*

tion Technology Security Conference Refereed R&D Stream (AusCERT), page 46, May 2006.

[28] P. England, Q. Shi, A. B. skwith, and F. Bouhafs. A survey of trust management in mobile ad-hoc networks. In *Proceedings of the 13th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking, and Broadcasting*. PGNET, 2012.

[29] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman. Network configuration protocol (NETCONF). *Internet Engineering Task Force, RFC*, 6241, 2011.

[30] L. Eschenauer, V. D. Gligor, and J. Baras. On trust establishment in mobile ad-hoc networks. In *Proceedings of the Security Protocols Workshop*, pages 47–66. Springer-Verlag, 2002.

[31] Y. Feng. Adaptive trust management in MANET. In *Proceedings of the IEEE International Conference on Computational Intelligence and Security*, pages 804–808. IEEE, 2007.

[32] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: a network programming language. In *Proceedings of the ACM SIGPLAN Notices*, volume 46, pages 279–291. ACM, 2011.

[33] Open Networking Foundation. SDN architecture overview, December 2013.

[34] T. Ghosh, N. Pissinou, and K. Makki. Collaborative trust-based secure routing against colluding malicious nodes in multi-hop ad hoc networks. In *Proceedings*

of the 29th Annual IEEE International Conference on Local Computer Networks, pages 224–231. IEEE, 2004.

[35] T. Ghosh, N. Pissinou, and K. Makki. Towards designing a trusted routing solution in mobile ad hoc networks. *Mobile Networks and Applications*, 10(6):985–995, 2005.

[36] E. Gray, P. Oconnell, C. Jensen, S. Weber, J. Seigneur, and C. Yong. Towards a framework for assessing trust-based admission control in collaborative ad hoc applications. *Technical Report of Computer Science Department, Trinity College Dublin*, 66, 2002.

[37] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4D approach to network control and management. *ACM SIGCOMM Computer Communication Review*, 35(5):41–54, 2005.

[38] G. C. Hadjichristofi, W. J. Adams, and N. J. Davis IV. A framework for key management in mobile ad hoc networks. In *Proceedings of the IEEE International Conference on Information Technology: Coding and Computing*, volume 2, pages 568–573. IEEE, 2005.

[39] Z. Han and W. Ren. A novel wireless sensor networks structure based on the SDN. *International Journal of Distributed Sensor Networks*, 2014:7, March 2014.

[40] Q. He, D. Wu, and P. Khosla. SORI: a secure and objective reputation-based incentive scheme for ad-hoc networks. In *Proceedings of the IEEE on Wireless Communications and Networking Conference*, volume 2, pages 825–830. IEEE, 2004.

[41] Y. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the 2nd ACM Workshop on Wireless Security*, pages 30–40. ACM, 2003.

[42] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks*, pages 135–147. ACM, 2003.

[43] EstiNet Technologies Inc. Estinet OpenFlow network simulator and emulator. *Communications Magazine, IEEE*, 51(9):110–117, 2013.

[44] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Openflow random host mutation: transparent moving target defense using software defined networking. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 127–132. ACM, 2012.

[45] T. Jiang and J. S. Baras. Ant-based adaptive trust evidence distribution in MANET. In *Proceedings of the 24th IEEE International Conference on Distributed Computing Systems Workshops*, pages 588–593. IEEE, 2004.

[46] N. Komninos, D. Vergados, and C. Douligeris. A two-step authentication framework for mobile ad hoc networks. *China Communications Journal*, 4(1):28–39, 2007.

[47] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, et al. Onix: a distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX Conference*

on Operating Systems Design and Implementation (OSDI), volume 10, pages 1–6, 2010.

[48] D. Kreutz, F. Ramos, and P. Verissimo. Towards secure and dependable software-defined networks. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pages 55–60. ACM, 2013.

[49] I. Ku, Y. Lu, and M. Gerla. Software-defined mobile cloud: architecture, services and use cases. In *Proceedings of the IEEE on International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1–6. IEEE, 2014.

[50] L. Lamport. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[51] H. Li and M. Singhal. Trust management in distributed systems. *Computer*, 40(2):45–53, February 2007.

[52] R. Li, J. Li, P. Liu, and H. Chen. On-demand public-key management for mobile ad hoc networks. *Wireless Communications and Mobile Computing*, 6(3):295–306, 2006.

[53] W. Li and A. Joshi. Security issues in mobile ad hoc networks-a survey. *Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County*, pages 1–23, 2008.

[54] X. Li, M. R. Lyu, and J. Liu. A trust model based routing protocol for secure ad hoc networks. In *Proceedings of the IEEE on Aerospace Conference*, volume 2, pages 1286–1295. IEEE, 2004.

[55] M. Liyanage, I. Ahmed, M. Ylianttila, J. Santos, and R. Kantola et al. Security for future software defined mobile networks. In *Proceedings of 9th International Conference on Next Generation Mobile Applications Services and Technologies*, 2015.

[56] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang. URSA: ubiquitous and robust access control for mobile ad hoc networks. *IEEE/ACM Transactions on Networking (ToN)*, 12(6):1049–1063, 2004.

[57] T. Luo, H. Tan, and T. Q. Quek. Sensor OpenFlow: enabling software-defined wireless sensor networks. *IEEE Communications Letters*, 16(11):1896–1899, 2012.

[58] J. Marshall, V. Thakur, and A. Yasinsac. Identifying flaws in the secure routing protocol. In *Proceedings of the IEEE International Conference on Performance, Computing, and Communications*, pages 167–174. IEEE, 2003.

[59] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 255–265. ACM, 2000.

[60] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L.Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[61] P. Michiardi and R. Molva. Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Advanced Communications and Multimedia Security*, pages 107–121. Springer, 2002.

[62] M. E. Moe, B. E. Helvik, and S. J. Knapskog. TSR: trust-based secure MANET routing using HMMs. In *Proceedings of the 4th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pages 83–90. ACM, 2008.

[63] M. Moloney and S. Weber. A context-aware trust-based security system for ad hoc networks. In *Proceedings of the IEEE Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks*, pages 153–160. IEEE, 2005.

[64] M M.Pushpalatha, R.Venkataraman, and T. Ramarao. Trust based energy aware reliable reactive protocol in mobile ad hoc networks. *World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, 3(8):1535–1538, 2009.

[65] J. Mundinger and J. Le Boudec. Analysis of a reputation system for mobile ad-hoc networks with liars. *Performance Evaluation*, 65(3):212–226, 2008.

[66] R. K. Nekkanti and C. Lee. Trust based adaptive on demand ad hoc routing protocol. In *Proceedings of the 42nd ACM Annual Southeast Regional Conference*, pages 88–93. ACM, 2004.

[67] E. Ngai and M. R. Lyu. Trust-and clustering-based authentication services in mobile ad hoc networks. In *Proceedings of the IEEE on Distributed Computing Systems Workshops*, pages 582–587. IEEE, 2004.

[68] B. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti. A survey of software-defined networking: Past, present, and future of programmable net-

works. *Communications Surveys and Tutorials, IEEE Communications Society*, 16(3):1617–1634, 2014.

[69] K. Paul and D. Westhoff. Context aware detection of selfish nodes in DSR based ad-hoc networks. In *Proceedings of the IEEE on Global Telecommunications Conference*, volume 1, pages 178–182. IEEE, 2002.

[70] K. Pentikousis, Y. Wang, and W. Hu. MobileFlow: toward software-defined mobile networks. *IEEE Communications Magazine*, 51(7), 2013.

[71] J. H. Perkins, S. Kim, S. Larsen, S. Amarasinghe, J. Bachrach, M. Carbin, C. Pacheco, F. Sherwood, S. Sidiroglou, G. Sullivan, et al. Automatically patching errors in deployed software. In *Proceedings of the 22nd ACM SIGOPS Symposium on Operating Systems Principles*, pages 87–102. ACM, 2009.

[72] A. A. Pirzada and C. McDonald. Establishing trust in pure ad-hoc networks. In *Proceedings of the 27th Australasian Conference on Computer Science*, volume 26, pages 47–54. Australian Computer Society, Inc., 2004.

[73] A. A. Pirzada, C. McDonald, and A. Datta. Performance comparison of trust-based reactive routing protocols. *IEEE Transactions on Mobile Computing*, 5(6):695–710, 2006.

[74] R. Ramanathan and J. Redi. A brief overview of ad hoc networks: challenges and directions. *IEEE Communications Magazine*, 40(5):20–22, may 2002.

[75] S. Reidt, S. D. Wolthusen, and S. Balfe. Robust and efficient communication overlays for trust authority computations. In *Proceedings of the IEEE on Sarnoff Symposium*, pages 1–5. IEEE, 2009.

[76] T. Roscoe and S. Rooney. Overview of the DCAN project. *Systems Research Group ATM Document Collection*, 4, 1995.

[77] L. Ruidong, L. Jie, L. Peng, and C. Hsiao-Hwa. An objective trust management framework for mobile ad hoc networks. *Proceedings of the 65th IEEE Vehicular Technology Conference (VTC)*, page 56, 2007.

[78] M. A. S. Santos, B. A. A. Nunes, K. Obraczka, T. Turletti, B. T. de Oliveira, and C. B. Margi. Decentralizing SDN's control plane. In *Proceedings of the 39th IEEE Conference on Local Computer Networks (LCN)*, pages 402–405, Sept 2014.

[79] K. Scarfone and P. Mell. Guide to intrusion detection and prevention systems (IDPS). *NIST special publication*, 800(2007):94, 2007.

[80] J. Sen, P. R. Chowdhury, and I. Sengupta. A distributed trust mechanism for mobile ad hoc networks. In *Proceedings of the IEEE International Symposium on Ad Hoc and Ubiquitous Computing*, pages 62–67. IEEE, 2006.

[81] P. K. Shanmugam, N. D. Subramanyam, J. Breen, C. Roach, and J. Van der Merwe. DEIDtect: Towards distributed elastic intrusion detection. In *Proceedings of the ACM SIGCOMM Workshop on Distributed Cloud Computing*, Aug 2014.

[82] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson. FRESCO: modular composable security services for software-defined networks. In *Proceedings of NDSS*, 2013.

[83] S. Shin, V. Yegneswaran, P. Porras, and G. Gu. AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. In *Proceedings of ACM CCS*, 2013.

[84] R. Skowyra, S. Bahargam, and A. Bestavros. Software-defined ids for securing embedded mobile devices. In *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*, pages 1–7. IEEE, 2013.

[85] S. Soltanali, S. Pirahesh, S. Niksefat, and M. Sabaei. An efficient scheme to motivate cooperation in mobile ad hoc networks. In *Proceedings of the 3rd IEEE International Conference on Networking and Services*, page 98. IEEE, 2007.

[86] Santashil S.PalChaudhuri and David B Johnson. Power mode scheduling for ad hoc networks. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pages 192–193. IEEE, 2002.

[87] Y. L. Sun, Z. Han, and K. R. Liu. Defense of trust management vulnerabilities in distributed networks. *IEEE on Communications Magazine*, 46(2):112–119, 2008.

[88] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, 1997.

[89] G. Theodorakopoulos and J. S. Baras. On trust models and trust evaluation metrics for ad hoc networks. *IEEE on Selected Areas in Communications*, 24(2):318–328, 2006.

[90] M. Virendra, M. Jadliwala, M. Chandrasekaran, and S. Upadhyaya. Quantifying trust in mobile ad-hoc networks. In *Proceedings of the IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS)*, pages 65–70. IEEE, April 2005.

[91] B. Wang, S. Soltani, J. Shapiro, and P. Tan. Local detection of selfish routing behavior in ad hoc networks. *Journal of Interconnection Networks*, 7(01):133–145, 2006.

[92] S. Wang, C. Chou, and C. Yang. EstiNet simulator and emulator. `http://www.estinet.com/`. Accessed: 2016-02-11.

[93] A. Weimerskirch and G. Thonet. A distributed light-weight authentication model for ad-hoc networks. In *Proceedings of the International Conference on Information Security and Cryptology*, pages 341–354. Springer, 2002.

[94] T. Xing, D. Huang, L. Xu, C. Chung, and P. Khatkar. Snortflow: A openflow-based intrusion prevention system in cloud environment. In *Research and Educational Experiment Workshop (GREE), 2013 Second GENI*, pages 89–92. IEEE, 2013.

[95] T. Xing, Z. Xiong, D. Huang, and D. Medhi. SDNIPS: Enabling software-defined networking based intrusion prevention system in clouds. In *Network and Service*

Management (CNSM), 2014 10th International Conference on, pages 308–311, Nov 2014.

[96] Z. Yan and R. MacLaverty. Autonomic trust management in a component based software system. In *Autonomic and Trusted Computing*, pages 279–292. Springer, 2006.

[97] Z. Yan, P. Zhang, and T. Virtanen. Trust evaluation based security solution in ad hoc networks. In *Proceedings of the 7th Nordic Workshop on Secure IT Systems*, volume 14, 2003.

[98] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang. Security in mobile ad hoc networks: challenges and solutions. *Wireless Communications, IEEE*, 11(1):38–47, 2004.

[99] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas. A quantitative trust establishment framework for reliable data packet delivery in MANETs. In *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 1–10. ACM, 2005.

[100] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas. Robust cooperative trust establishment for MANETs. In *Proceedings of the 4th ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 23–34. ACM, 2006.