UNIVERSITY OF TEXAS AT ARLINGTON

MASTERS THESIS

A Survey on DDoS Attacks in Edge Servers

Author:
Iftakhar AHMAD

Supervisor:
Dr. Mohammad Atiqul ISLAM

A thesis submitted in fulfillment of the requirements for the degree of Masters of Computer Science

in the

Computer Science and Engineering Department

December 8, 2020

UNIVERSITY OF TEXAS AT ARLINGTON

Abstract

Computer Science and Engineering Department

Masters of Computer Science

A Survey on DDoS Attacks in Edge Servers

by Iftakhar AHMAD

In modern times, the need for latency sensitive applications is growing rapidly. Cloud computing infrastructure is unable to provide support to such delay sensitive applications. Therefore, a new paradigm called edge computing has emerged. In edge computing various paradigms like Fog, Cloudlet, Mobile Edge Computing, etc. provide real-time, location aware services to users. As a result number of requests are generated for processing in the edge servers. If these edge servers for some reason become unavailable for providing service, users will not be able to perform their delay sensitive or location aware operations. Like other servers in the network, edge servers can also become vulnerable to various security attacks. Among other security attacks, Distributed Denial of Service (DDoS) attack can be very threatening for edge servers. Capacity of edge servers compared to traditional servers is limited. Therefore, the effort needed to overwhelm the edge server will be smaller, on the other hand, the impact of DDoS attacks on edge servers will be dreadful. Importance of a proper functioning of edge servers is immense for providing uninterrupted support to the users. Therefore, some kind of recovery methods is needed to recover an overloaded edge server. Present state-of-art mentions some such recovery methods for the edge servers so that they can provide continuous services.

Acknowledgements

I would like to express my heartfelt gratitude to my supervising professor Dr. Islam, for his persistent support and guidance through out this research work. Whenever I was confused or stuck at some point he helped to overcome that. He guided me with his experience to successfully complete this research work.

I would also like to express my gratitude to rest of my committee members Dr. Ming Li and Dr. David Levine for providing valuable suggestions and feedback which helped me to improve my research work.

I would like to thank committees of the department of Computer Science and Engineering for providing me financial support which helped me focus in my study and research work.

Finally, I would like to express my gratitude to my parents, wife and sister who provided me mental support when it was needed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's world influence of cloud computing is enormous in day-to-day life. Cloud computing has made many recent technologies possible and feasible. Many real-time applications depend on cloud computing. Emergence of IoT also, to some extent, rely on cloud computing. However, cloud computing has some drawbacks like long latency due to long distance from user. To mitigate the limitations of cloud computing, a new technology called edge computing is introduced. In edge computing, edge server, where all the processing is done, is placed near the users. As a result the distance between the edge server and user is reduced and latency is decreased. This edge computing is suitable for latency-sensitive applications. Latency-sensitive applications are growing their importance day by day. Augmented Reality, Virtual Reality, autonomous vehicles etc. are some examples of latency-sensitive applications. These applications cannot allow delay while communicating. Therefore, if they have to depend on distant cloud servers there would be delays in communications that cannot be tolerated. Edge computing comes into the picture here. Edge computing brings the task processing servers to the proximity of users using the likes of above mentioned applications. Therefore, end-to-end delay is reduced and proper execution of such applications is made possible. Due to ever-growing number of real time, delay-sensitive applications in recent times, the edge computing has emerged as a field of utmost importance. There are several categories of edge computing paradigm. Fog computing [81], mobile edge computing [7], mobile cloud computing [82] etc. are examples of such categories of edge computing paradigm. We will discuss in detail these categories of edge computing in the next section.

Security is important for all these classifications of edge computing paradigm like any other entities in the computer network. There are lots of security threats for entities in computer networks. In the same way there are lots of ways for mitigation of many of these security threats. If there is an entity in the network, there will be security vulnerabilities for it. Current state-of-art literatures provides mitigation strategies for most of the existing vulnerabilities. However, still there are some unsolved security threats. For edge computing paradigm there are some security threats that are common to other entities in the network also, however, some new security threats also exist for them. Since idea of edge computing comes from cloud computing, edge computing inherits some of security threats from cloud computing. There are lots of security threats for cloud computing [34]. Out of these threats some are also found in edge computing. In edge computing main functional part is edge server where all computations are performed. This edge server is mainly the target of various security attacks. Denial of Service attack, Man in the Middle attack, Rogue Gateway etc. is some of the possible security attacks in edge servers. There are some existing remedies for these attacks. However, all the remedies may not be suitable for edge computing paradigm. We will discuss more about security attacks and their remedies in later sections.

DDoS or Distributed Denial of Service attack can be a foreseeable threat for any organization on the internet. In recent past DDoS attack is one of the most prominent security vulnerabilities for web entities. The prime target of DDoS attack is exploit various security vulnerabilities and launch an attack to a target server and exhaust its all resources. In this way server becomes unavailable to legitimate users. DDoS attacks not only hampers usual operation of a server but also destroys the reputation of the corresponding organization. DDoS attacks also cause economic loss for the target organization. Edge servers usually provide latency-sensitive computations into some of the real-time applications like virtual reality, augmented reality, etc. This kind of server also provides support for task offloading from the devices with limited computation capability. Therefore, if these servers are made unavailable to legitimate users, they will not be able to perform delay sensitive operations and provide support for task offloading. This is the motivation for attackers to target edge servers for DDoS attacks. Moreover, edge servers do not have very rich resources for performing their computations. This makes the edge servers an easy prey for attackers. Because attacker will not require much resources to take down an edge server through DDoS attack. This survey illustrates a roadmap on how to launch a DDoS attack to an edge server by reviewing existing literatures.

The main contributions of this survey are:

1. Paves the way for future researchers interested in security of edge servers.
2. Illustrates how to launch DDoS attacks in edge servers.
3. Providing an intuitive idea on how DDoS attacks can be mitigated in edge servers.

Rest of the paper is organized as follows: Overview of different edge computing paradigms are discussed in section 2. Next different aspects of DDoS attacks are discussed in section 3. Computations offloading is a major aspect in edge computing which is discussed in section 4. Particulars of how to launch a DDoS attack targeting an edge server is discussed in section 5. Finally, this survey concludes with summary and conclusion in the sections 6 and 7 respectively.

Chapter 2

Overview of Edge Computing Paradigms

2.1 Fog Computing

Fog computing is a platform that brings the cloud computing to the proximity of end users. The term "Fog" was introduced by Cisco and is analogous with real-life fog [9]. Cloud is at a larger distance on the contrary fog is closer to earth. In a similar way cloud computing is at a larger distance from earth and Fog computing is closer to earth. However, cloud and fog both share similar set of services, for example computation, storage, and networking. Nonetheless, some differences are present between these two. Specific geographic regions are targeted in Fog computing. Moreover, the fog is suitable for applications which need real-time response and low latency, for example interactive and IoT applications. On the other hand, the cloud is a centralized entity and is far from the user therefore, it has some performance limitations with respect to latency and response time for real-time applications [8]. The fog computing does not disown cloud computing rather than complements it. Fog architecture makes the creation of a hierarchical infrastructure possible. In Fog computing all the analysis of local information is performed at the 'ground', and the co-ordination and global analytics are performed at the 'cloud'. Here, cloud services are deployed mostly at the edge of the network, however, these calculations can also be performed in other locations, for example, IP/multiprotocol label switching (MPLS) backbones. The fog network infrastructure is heterogeneous, the high-speed links and wireless access technologies coexist here [72]. As depicted in Fig. 2.1, a multi-tiered architecture is necessary. The first part comprises of

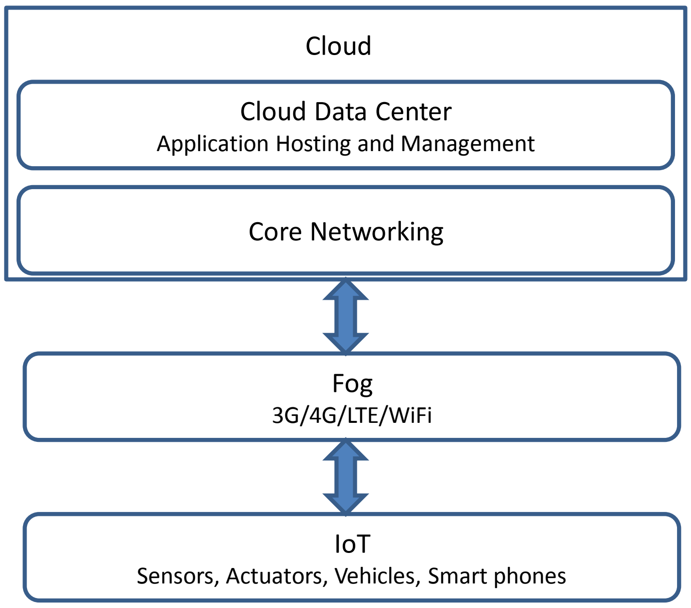

Cloud
Cloud Data Center
Application Hosting and Management
Core Networking
Fog
3G/4G/LTE/WiFi
IoT
Sensors, Actuators, Vehicles, Smart phones

FIGURE 2.1: Three layer architecture consisting of cloud fog and IoT devices [8]

IoT applications running in the end user devices say, a vehicle. Fog computing is the second tier of this multi-tier architecture which is connected with end users through a router, access point, wireless access network, or an LTE base station. Cloud's data center is the last tier of this 3-tier architecture. Due the 3-tier architecture, the IoT applications and services can be operated from edge of the network as allowed by Fog computing, as well as from end devices, like gateways, routers, access points, Road Side Units (RSUs), and Machine to Machine (M2M) gateways [19, 20]. Moreover, Fog is allowed to perform real-time monitoring, actuation, data analysis with reduced latency, improved QoS, and saving of bandwidth as data are processed at the edge of the network [8]. In the above mentioned multi-tier architecture the centralized cloud servers and fog coexist with each other. However, it is not mandatory that Fog needs cloud servers to perform its tasks, rather it can function independently [55]. As a result Fog can facilitate lots of applications some solely by itself and some others taking help from centralized cloud servers.

Like other entities on the internet, Fog computing also has some limitations and challenges. Fog computing inherits some of the challenges from cloud computing. Moreover, it has some challenges of its own. Some of these challenges are heterogeneous devices management, architectural issues, mobility, security, and privacy issues. There are heterogeneous devices in Fog computing. When there are heterogeneous devices, interoperability among them becomes a challenging task. Scalability issues may arise when many devices are connected. For proper management of resources and load balancing, we need an efficient resource scheduler. It is very challenging to design such resource scheduler. Moreover, it is demanding to ensure proper monitoring and management of connected devices. It is more crucial for devices with real-time applications. Monitoring of real-time traffic and billing mechanism is needed for Fog computing. Devising a fair billing model for the services offered is challenging. The billing model has to be fair to attract subscribers and generate good revenue. However, availability of any standard billing model for Fog computing is still an open research issue. Moreover, it is key challenge to protect Fog computing from malicious attackers and security threats [8].

2.2 Cloudlets

Cloudlets were first developed by a team at Carnegie Mellon University [75]. Similar to Fog computing, in the 3-tier architecture cloudlet represents the middle tier i.e. 3-tier architecture is mobile device – cloudlet – cloud. Various cloud services are provided by Cloudlets to number of mobile users within the coverage range of their Wi-Fi access point. However, there is an increasing demand of the mobile applications with high computing and storage capabilities with free user mobility which makes Cloudlets an inefficient solution for end user task offloading. Whereas, task offloading is a key feature in edge computing paradigm. The reason for Cloudlets being inefficient solution are the limited resource capabilities of the Cloudlet servers and the short coverage range of the Cloudlet Wi-Fi connection [48]. Formally we can define a Cloudlet as a cluster of trusted computers, which are well connected to the Internet, which have resources available to nearby mobile devices [75]. We can treat a Cloudlet as a ”data center in a box” which is running a virtual machine that is capable of provisioning its resources to users and end-devices in real time over a WLAN network. Cloudlets provides services over a one-hop access with high bandwidth, and offers low latency for applications [26]. Cloudlets mainly consist of a cluster of resource-rich and multi-core computers. These computers are equipped with high-speed internet connection and high bandwidth wireless LAN to be used by neighboring mobile devices. Cloudlets are stored in a tamper-proof box for ensuring safety and security in an unmonitored area [75]. In present world mobile devices are much sophisticated. However, still these devices are not competent with other related technologies like laptops and servers. On the other hand latest mobile applications are resource hungry. To run these resource hungry applications on mobile devices with limited resource capacity Cloudlets come in handy. Cloudlets virtualize various features provide computational resources to the users of mobile devices. The mobile device which acts as a thin client, offloads computational tasks to a Cloudlet through a wireless network. The presence of a cloudlet in mobile device's proximity is necessary, so that end-to-end response time with executing applications remains smaller and predictable. There is a basic difference between cloud and a cloudlet, in a cloudlet only the soft state of data or code is stored, on the contrary, a cloud stores both soft and hard states of the data or code. As a result a cloudlet's failure does not result in any data loss of mobile devices [8]. Authors in [75] developed a prototype of a cloudlet, which is named as Kimberly. The cloudlet infrastructure includes a desktop computer running Maemo 4.0 Linux. The mobile device which is used in the prototype is Nokia N810. The mobile user in this prototype utilizes VM technology for instantiating a service on an adjacent cloudlet and uses wireless LAN to interact with this cloudlet.

2.3 Mobile Edge Computing

Mobile edge computing is another paradigm of Edge computing. To describe execution of services at the edge of the network the term Mobile Edge Computing (MEC) was first introduced in 2013. At this time IBM and Nokia Siemens Network launched a platform which could run applications within the range of a mobile base station [46]. To enable flexible and quick deployment of new services and applications, MEC servers are placed at the vicinity of the users i.e. at cellular base stations. In MEC we can consider servers like cloud servers run at the edge of mobile networks. The servers in MEC perform tasks that cannot be achieved with traditional cloud network infrastructure due to limitations like long execution delays. MEC directs the traffic targeted for the cloud to the MEC servers rather than forwarding all the traffic to the remote cloud servers. Therefore, applications are run and related processing tasks are performed at the proximity of cellular customers. This process reduces both network congestion and response time of applications. In this procedure a request from user is either served from the MEC server and thereby response can be quickly delivered to end users or sometimes the user request can be forwarded to a remote cloud [8].

Dated in 2014, MEC obtained its present meaning. At that time the Industry Specification Group (ISG) of Mobile-Edge Computing was released by ETSI [31]. According to this specification the purpose of MEC is to "provide an IT service environment and cloud-computing capabilities at the edge of the mobile network". This specification group proposes creation of an open ecosystem, where the applications can be deployed by service providers over a multi-vendor MEC. When the formation of the standard is done, it is telecommunication companies who will be in charge of deploying the MEC service environment in their established infrastructure. Low latency, high bandwidth, and access to radio network information and location awareness are some of the benefits achieved when cloud services are deployed at the edge of mobile networks like 5G [72]. As a result of this, it becomes viable to optimize the existing mobile infrastructure services, or even in some cases implement novel ones. An example of such an application is the Mobile Edge Scheduler [33]. In this application the mean delay of general traffic flows in the LTE downlink is minimized. Moreover, mobile network operators will not be solely beneficiaries of the deployed services. However, 3rd party service providers will also be able to use the services as well. Augmented reality, intelligent video acceleration, connected cars, and Internet of Things gateways, are some of the prime applications which can be implemented using this service [45].

This promising paradigm MEC, has some downside. This downside is related to the management complexity of large scale deployments which offers many applications served to millions of users. MEC accelerates different types of applications and also provides smooth data streaming in mobile networks. MEC makes this possible through caching of application and streaming data. It also augments the the processing capabilities of mobile end users with the help of its processing capabilities at the edge of the mobile network. According to its basic principle in MEC a decentralized framework is suggested rather than depending on a centralized platform, to serve the necessities of all mobile devices. This procedure can be obtained by transferring some of the cloud servers to the edge of the mobile network [49]. According to authors in [21], this decentralized platform was first proposed by Akamai Technologies who used their content delivery network to describe the topology. In this topology frequently requested contents are cached at MEC which is located at the network edge. In recent studies it was proposed that integration of MEC and cloud computing servers makes it possible to serve more complex services by reducing the load on centralized cloud, by avoiding bottlenecks and single points of failure.

2.4 Mobile cloud computing

The main focus of Mobile Cloud Computing (MCC) is the notion of 'mobile delegation'. Mobile devices have limited resources, therefore, they need to delegate the large storage of data and computationally intensive tasks to powerful remote servers. According to original MCC concept which was introduced in 2009, centralized cloud computing platforms were only suitable solution for executing tasks remotely [1]. Later on the scope of MCC was expanded by other researchers. According to this new vision, devices located at the edge of the network were made available for executing tasks remotely [5]. Right now, both these visions of MCC coexist with each other [67].

At the beginning, MCC was designed to provide several novel solutions to various services including mobile learning, mobile healthcare, searching services etc. [23]. At present, some of the above mentioned services can be performed in a centralized cloud or in the corresponding mobile devices. However, the notion of MCC is still pertinent, as the potential of MCC is yet to be fully exploited. There are some kind of applications, for example, augmented reality and augmented interface applications, for which the MCC architecture can be advantageous. Several advantages such as smaller delay and access to context information can be achieved when a execution platform is available at proximity of the mobile devices [82].

In MCC, one of the most active research areas is how tasks can be delegated to external services [67]. To utilize the advantages to MCC there are many applications which allow their code to be migrated from mobile devices to cloud-based options present at the network edge. .NET and JVM like frameworks are used to build these applications and make it easier to migrate to code. According to some research mobile devices are allowed to migrate only some part of their code. However, other researchers suggest migration of an entire execution environment i.e. a clone of the mobile device to the MCC. After that, memory image, CPU state and some other parts of the mobile application are added into the clone. Finally, mobile agent infrastructures are used in some approaches, where a mobile agent is created by the mobile device which acquires or processes information on its behalf [72].

Chapter 3

Overview of DDoS Attacks

In this section we will discuss the overview of DDoS attacks and its mitigation process as described in state-of-art literature.

DDoS attack can be formally defined as below:

Distributed Denial of Service (DDoS) attacks, according to the WWW Security FAQ [80] can be defined as "A DDoS attack uses many computers to launch a coordinated DoS attack against one or more targets. Using client/server technology, the perpetrator is able to multiply the effectiveness of the DoS significantly by harnessing the resources of multiple unwitting accomplice computers, which serve as attack platforms". The most dangerous and advanced form of DoS attacks is the DDoS attack. DDoS attack is differentiated from other security attacks by deploying its attack weapons in a "distributed" manner and to accumulate these forces to generate a deadly traffic. Traditional security defense techniques are not sufficient for DDoS attacks because it does not break the victim's system. The main purpose of a DDoS attack is to harm a victim either for personal reasons, or for some sort of material gain, or for gaining popularity [27].

A Distributed Denial of Service Attack comprises of four basic elements, which are shown in Fig. 3.1. These elements are discussed in short below.

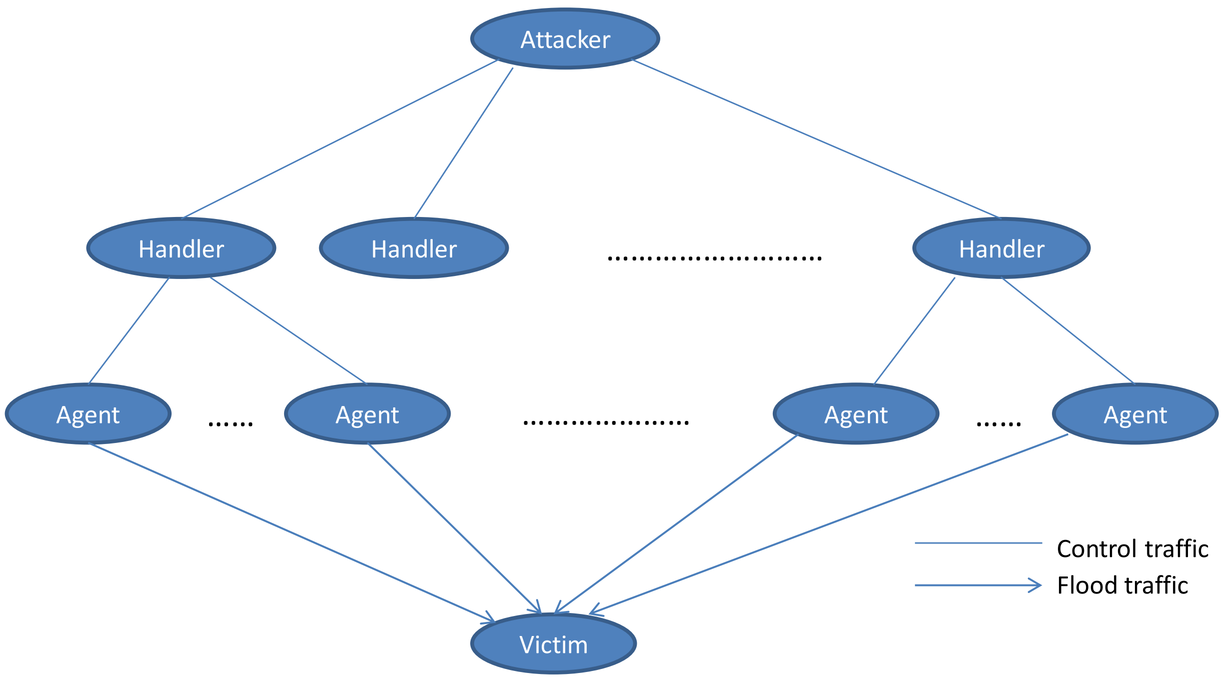


FIGURE 3.1: DDoS attacks architecture [27]

- The actual attacker.
- The handlers or masters. They contain malicious hosts within which special programs are running. They are capable of taking control over multiple agents.
- The attack daemon agents or also known as zombie hosts. They are malicious of compromised hosts in which a special program is running. They also generate a packets' stream

targeting the victim. These machines are usually outside of the victim's network. As a result efficient response from the victim can be avoided. They are also outside of the attacker's network. Therefore, liability can be avoided if the attack is traced down.

- An actual victim or target host.

DDoS attack is a kind of coordinated attack, where a large number of compromised and distributed devices are deployed to execute the attack. The attacker infects the bots with the help of malware and brings them under the his control. Then these bots are directed by the attacker and they send attack packets to victims [44, 51, 52]. In DDoS attack attacker sends a large volume of attack traffic, for example, the attacks like TCP SYN flooding, UDP flooding, and ICMP flooding and overwhelms the network infrastructure including servers, routers and switches [28]. Sometimes in DDoS attack the attack traffic is similar to legitimate traffic. Moreover, protocol vulnerability is exploited to consume an excessive amount of resources for example HTTP protocol is targeted with an intention to exhaust resources of the server which was made available for Web services [88, 59].

One key issue of DDoS attack and its defense is competition of available resources. When a defender has enough resources to beat a DDoS attack, the attacker cannot be successful, and vice versa. Cloud computing is a popular target for DDoS attacks. Unfortunately, the components of clouds, for example, the server-client component and the peer-to-peer computing platforms, lack required resources to beat a DDoS attacks. However, a large amount of resources is pooled by a cloud infrastructure provider and easy access is provided to them. In this way cloud can handle a rapid increase in demand of service [35]. As a result, it is not possible to shut down a cloud through a DDoS attack. Although cloud infrastructure cannot be blocked by a DDoS attack, individual cloud customers (referred to as parties hosting their services in a cloud) cannot avoid the damages of a DDoS attack as they usually do not have the advantage [87].

Internet based applications such as independent news web sites, e-business and online games require smooth network accessibility. Therefore, DDoS attack is vital threat to these kinds of applications [66]. Nowadays botnets are used to launch DDoS attacks. It is a long-held belief that as many computers or devices as needed can be easily compromised by an attacker. However, recent researches [32] have corrected this belief. The number of active bots, a botmaster can manipulate limited because of presence of anti-virus and anti-malware software installed in the target devices, although the amount of bot footprints may be much higher [87].

The objective of DDoS attacks is to take control over the network bandwidth so that legitimate users of the victim systems do not get the appropriate service. As this type of attack was observed by many studies and different schemes were proposed by them to protect the network and equipment from DDoS attacks, it is now more difficult than past to launch such attacks based on network layer. Due to failure of Net-DDoS attacks, the attacker moves their concentration to application-layer attacks and create a more advanced type of DDoS attacks. To avoid being detected, the attackers send HTTP GET requests to launch attack on the victim Web servers and lots of large image files are pulled from the victim server to overwhelm them [84].

Main reason why the rate of DDoS attacks has escalated notably is the use of botnets to carry out a DDoS attack. Botnets can be defined as networks which are formed by compromised bots or machines. Srizbi, Kraken/Bobax, and Rustock like large-scale botnets have gained popularity for conducting DDoS attacks. To infect a large number of machines in a short span of time in traditional networks is quite complex. However, attackers may exploit the on-demand self-service capabilities of the cloud and generate a powerful botnet instantly. Cloud computing makes malware-as-a-service possible and enables it to be used for launching DDoS attacks. The cost of malware-as-a-service has been decreasing quickly due to competition among the suppliers. This low cost makes it easier to launch large-scale DDoS attacks using botnets [69].

The target of DDoS attacks is to drain the resources including bandwidth, buffer, battery or processing unit by sending huge number of packets destined to the victim. Impact of generation of such unnecessary packets overwhelms the victim's resource capability. This kind of flooding of packets reduces victims working ability and also reduces the lifetime of the victim. Fig. 3.2 depicts a block diagram of such DDoS attack.

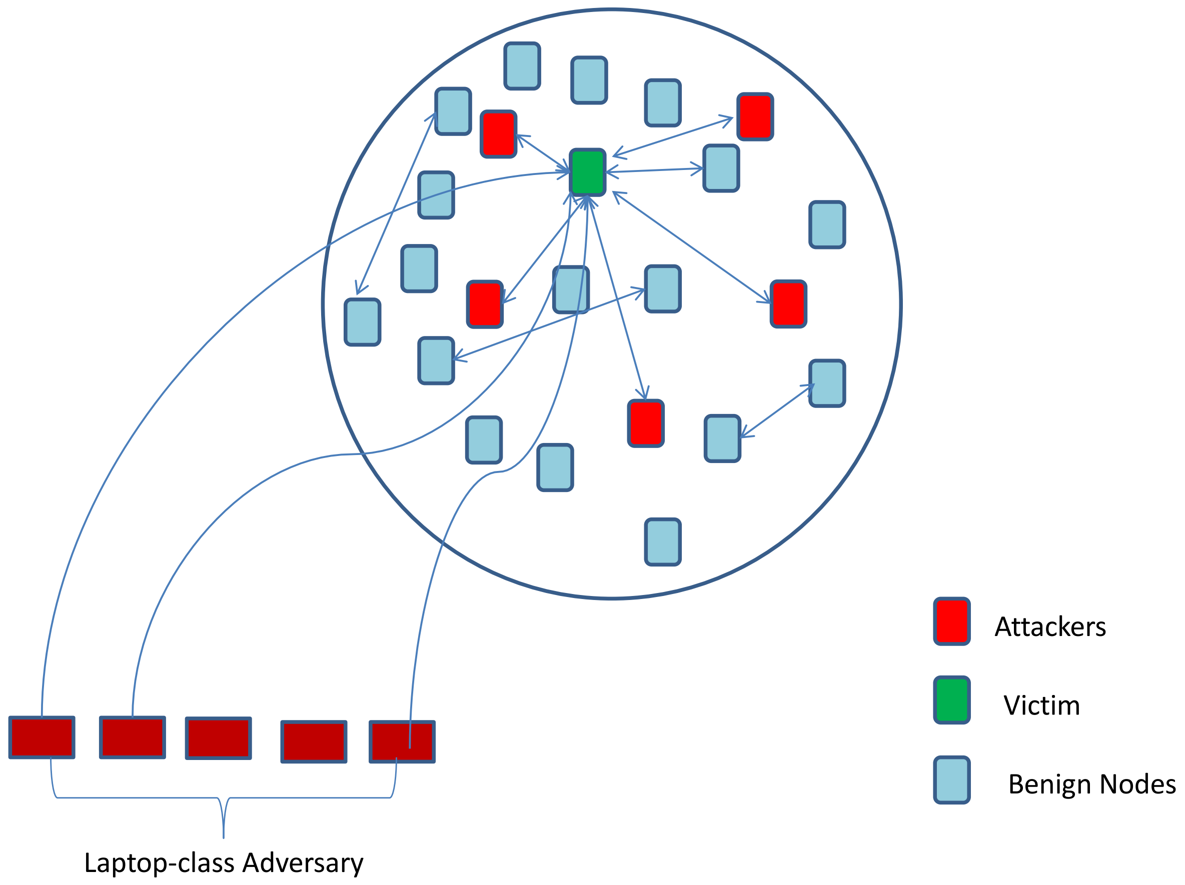


FIGURE 3.2: DDoS attack block diagram [38]

3.1 Classification of DDoS Attacks

Depending on the nature of DDoS attacks, it can be classified into three categories [76]. These three categories are discussed in short here:

1. Application DDOS Attack: One of the slow and low profile methods to deploy a DDoS attack is Application DDoS Attack. GET or POST method of HTTP protocol is used in order to drain or drop services. To do so server or software vulnerabilities are exploited in this kind of attack. This sort of attack mostly targets Apache or Open BSD servers and crash the server execution with successful attack. In this kind of approach, cloud computing applications are most suitable targets.

2. Protocol DDOS Attack: Target of Protocol attacks is heavy consumption of resources and intermediate layers for example firewall or HTTP protocols. To exhaust capabilities SYN floods, Ping of Death, Smurf DDOS, fragmented packet attacks etc. are used to implement protocol DDOS attack.

3. Volume DDOS Attack: The core concept of this approach is heavy Packet bombarding or flooding in order to exhaust the bandwidth or network resources. UDP and ICMP packets are used to absorb the resources and heavy packet flooding is done to generate congestion in the communication link [38].

A group of malicious nodes is required in the same network to launch a DDoS attack. [] states that this kind of requirement can be met by a zombie network. Intended services can be disabled and also heavy flooding can be done by engaging zombie nodes and goal of a DDoS attack can be achieved [70, 42].

3.2 Botnet Based DDoS Attacks

A network of machines is compromised by botnets. These machines are referred to as a bot, zombie, or drone. They are implemented under a command and control management infrastructure. Botnets management usually affects a number of systems using numerous tools and by the installation of a bot. This bot can remotely take control over the victim machine using the Internet relay chat (IRC) [11]. Main target of present botnets is frequently spreading DDoS attacks mainly on the Web. Moreover, while the bots are created, the communication approach can be changed by the attackers. Most of the bots vary their potentials in order to take part in such attacks. HTTP/S flooding attack is the most commonly generated botnet attack on the application layer. Such attacks launch bots made by the HTTP server. Therefore, this kind of bots is called "Web-based" bots [36].

Causing harm at the victim side is the main objective of a botnet based DDoS attack. Usually the ultimate target behind this kind of attack is personal. Therefore, blocking the available resources or degrading the performance of the service at the victim side which is needed for the victim machine is the target of such attack. As a result, the DDoS attacks are carried out for revenge purposes. Sometimes these sorts of attacks are launched to become popular in the hacker community. Moreover, these kinds of attacks also can be performed for the material gain by breaking the confidentiality and using the data for unauthorized use [2].

3.2.1 Categories of Botnet Based DDoS Attack Networks

There are three categories of DDoS attack networks which are botnet based. They are the agent-handler, IRC-based, and Web-based models.

Agent-Handler Model: Clients, handlers, and agents are three components of the agent-handler model of a DDoS attack as depicted in Fig. 3.3 In the DDoS attack system the attacker makes contact with the client. The handlers are various software packages which are located all over the Internet. These software packages are used by the client to contact with the agents. The agent software develops in a compromised system and at the appropriate time ultimately conducts the attack. The attacker contacts with the handlers in order to discover operational agents. Moreover attacker communicates with handlers to fix when to attack or to upgrade agents. Owners and users of agent systems usually cannot determine that their system is now compromised and is being used for DDoS attacks. The agents can communicate with one or multiple handlers at a time depending on the DDoS attack network configuration. A compromised router or network server is often chosen by attackers for installing handler software. Typically a large volume of traffic is handled by the target. This makes the message identification troublesome between the client and the handler and between the handler and the agents. In some descriptions of DDoS tools, the two terms "agents" and "handler" are replaced by "demons" and "master" respectively [79].

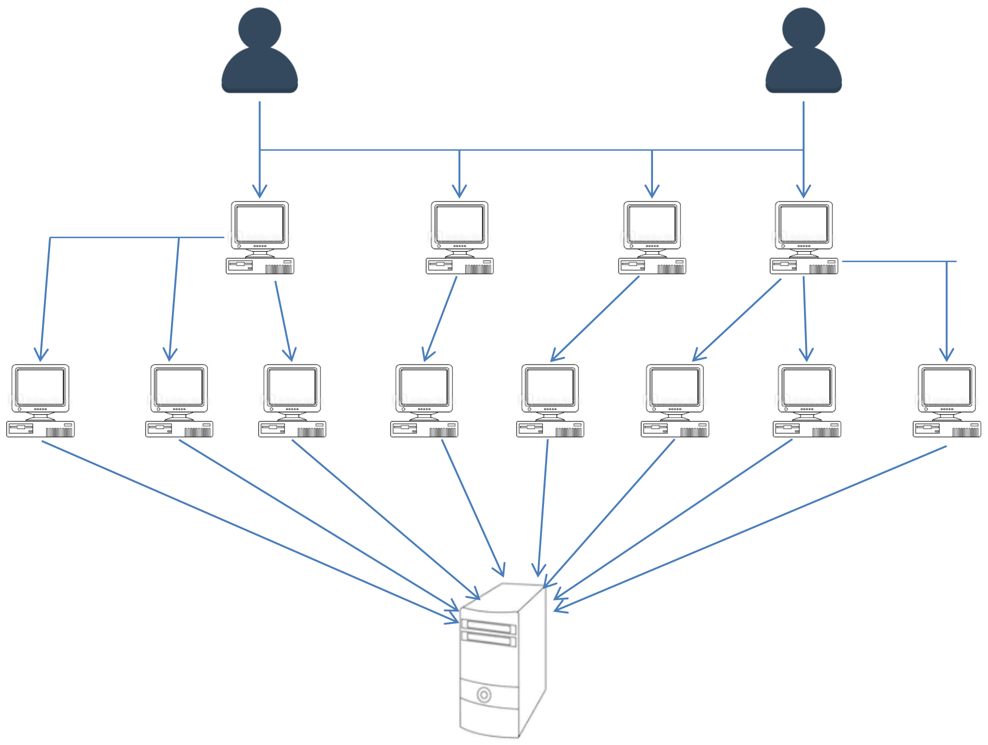


FIGURE 3.3: Agent handler model [2]

Internet Relay Chat (IRC) Model: Fig. 3.4 shows the IRC-based DDoS attack architecture. This model is similar to Agent handler model. However, in this model a handler program is not employed which is installed on a network server, rather an IRC communication channel is used to establish between the client and agents. In this model attacker has an advantage that "legitimate" ports can be used to send commands to agents. As a result tracking DDoS command packets is hindered. Moreover, usually, there is a large volume of traffics in the IRC servers. Therefore, attacker can easily hide their presence. The attacker can easily and momentarily access IRC server and access all available agents. Therefore, it does not need to maintain the list of the agents [54]. In the IRC network the agent sends and receives messages through IRC channel and when an agent becomes operational, it sends updates to the attacker.

Web-based Model: IRC-based model is mostly preferred for botnet command and control. However, in recent years Web-based reporting and command have emerged. In the Web-based model some bots just report the statistics to a Web site. On the other hand other bots are expected to be fully configured. Moreover, they are managed and controlled by complex PHP scripts. They are also responsible for encrypted communications over the 80/443 port and the HTTP/HTTPS protocol [2].

3.2.2 Botnet Based DDoS Attack Tools

There are lots of DDoS attack tools available and their architectures are very similar to each other. Therefore, there exist some tools which originate through small modifications of other tools [40].

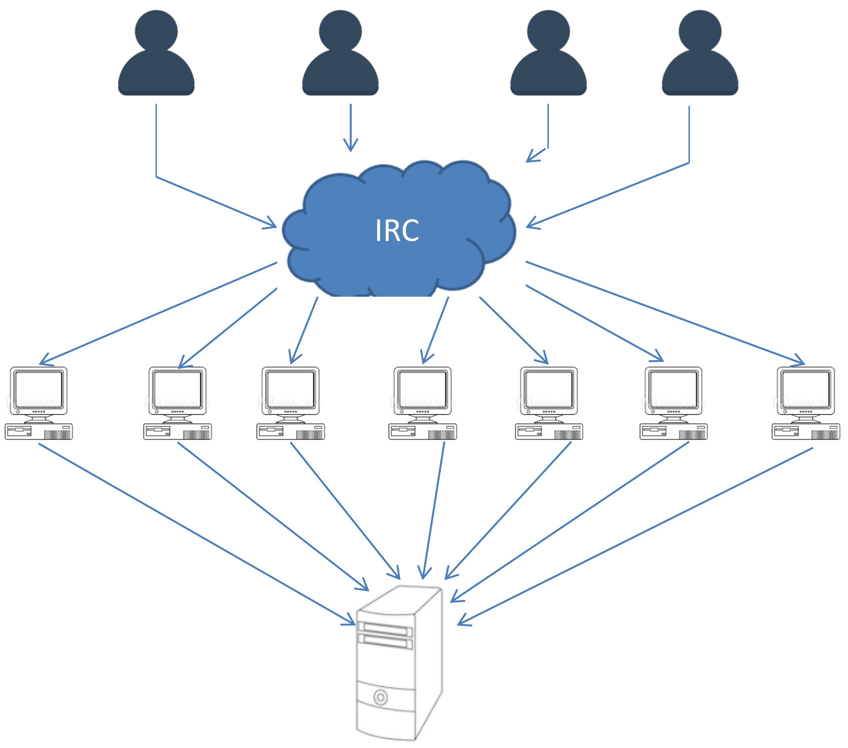


FIGURE 3.4: Internet Relay Chat Model [2]

Similar to categories of botnet based DDoS attack network the Botnet based DDoS attack tools are categorized as agent-based, IRC-based, or Web-based DDoS attack tools.

Agent-based DDoS Attack Tools: DDoS attack tools which are Agent-based, are constructed on the basis of the agent–handler DDoS attack model comprising handlers, agents, and victims. Trinoo, Tribe Flood Network (TFN), TFN2K, Stacheldraht, Mstream, Shaft etc. are examples of agent-based DDoS tools [39]. Trinoo [17] is most popular and widely used among the above mentioned agent-based DDoS tools. Because it has the capability of depleting bandwidth. Moreover, it has the ability to launch UDP flood attacks against one or number of Internet protocol (IP) addresses. Shaft [22] can launch packet flooding attacks as a result it is similar to Trinoo. Both the duration of the attack and the size of the flooding packets can be controlled in Shaft. Like Trinoo, TFN [24] is a DDoS attack tool which is able to launch bandwidth and resource depletion attacks. TFN is capable of doing Smurf, ICMP echo request flooding, TCP SYN flooding, UDP flooding, and ICMP directed broadcast. TFN2K [22] is able to launch Smurf, SYN, UDP, and ICMP flood attacks, which is a derivative of TFN. Encrypted messages can be added between attack components in TFN2K. Stacheldraht [25] came into existence after TFN attempts. Stacheldraht improves over a number of TFN's weak points and is able to implement Smurf, UDP flood, SYN flood, and ICMP flood attacks. On the contrary, Mstream [37] is a point-to-point TCP ACK flooding tool where fast-routing routine tables in some switches can be overwhelmed.

IRC-based DDoS Attack Tools: IRC-based DDoS attack tools were created after the growth of agent–handler attack tools. In recent past, sophisticated IRC-based tools have been developed. Important features of several agent-handler attack tools are included in these tools. One of the most well-known IRC-based DDoS tools is the Trinity, which is on top of UDP, TCP ACK, TCP SYN, and TCP NUL packet floods. TCP random flag packet floods, TCP fragment floods, TCP established floods, and TCP RST packet floods are introduced by the Trinity v3 [43]. myServer [22] was developed after the development of the Trinity. myServer depends on external programs to launch DoS and plague to simulate TCP ACK and TCP SYN flooding. Another light-weight and powerful IRC-based DDoS attack tool is Knight [12]. It can launch UDP flood attacks and

SYN attacks. Knight is considered to be an urgent pointer flooder [79]. Kaiten [12] is an IRC-based DDoS tool based on Knight. Kaiten carries out UDP, TCP flood attacks, SYN, and PUSH+ACH attacks.

Web-based DDoS Attack Tools: To launch attacks at the application layer, especially the Web server Web-based DDoS attack tools were developed. To launch attacks a Web server IRC-based DDoS attack tools with the HTTP/S flooding function are used. This proves that various new tools are being used by the attackers to conduct DDoS attacks. There are some downloadable tools and commercial services which can launch DDoS attacks free of cost [57]. There are more than 20,000 infected computers having multiple targets are able to destroy around 90 percent of Internet sites [68]. Calling someone in the world from one Website and a DDoS attack on the application layer are highly similar. While the Web site is being out of service (due to attack). As a result, the server in which the site is hosted cannot process all requests. On the contrary, the compromising handler which injects the site with bots that are controlled by the attackers. Different tools are used by the attacker to launch a successful attack. BlackEnergy [62], Low-Orbit Ion Cannon [58] and Aldi Botnet [77] are some example of web-based attack tools

3.2.3 Classification Of Botnets Based DDoS Attacks

Every day new kinds DDoS attacks are emerging. Moreover, some attacks remain undiscovered. Vulnerability exploitation is major aspect for various types of DDoS attacks. Consumption of the the host's resources characterizes the first type of attack. Usually a Web server or a proxy connected to the Internet becomes the victim. If the traffic load becomes high, senders are notified by sending packets. The senders can be either legitimate users or an attack source, are informed to reduce their sending rates. In response to such packets legitimate users decrease their sending rates, on the other hand attack sources increase or maintain their packet sending rates. As a result, the host's resources such as the CPU or memory capacity, depletes. Moreover, the host has to deny services to the legitimate traffic. The second kind of attack is based on network bandwidth consumption. Traffic from legitimate sources in the network gets obstructed if the network channel is mostly occupied by the malicious traffic. In reality, bandwidth DDoS attacks are more dangerous than resource consumption attacks [60]. These attacks discussed in short below:

Net DDoS-based Bandwidth Attacks: Net DDoS-based bandwidth attacks are usually launched from a single attack source which exploits specific IP weaknesses. There are various attacks of this type. Some examples include SYN Flood Attacks [83], ICMP Flood Attacks [89].

App-DDoS Attacks: Attack volume can be increased if the target is forced to execute costly operations. All the available bandwidth can be used by these types of attacks. Moreover, in this kind of attack the pipes are filled with illegitimate traffic. This sort of attacks hamper routing protocols and disrupt ongoing services due to resetting the routing protocols or by offering data which cause harm to the server operation [60]. Examples of such attacks include HTTP Flood Attacks [4], Session Initiation Protocol (SIP) Flood Attacks [47], Distributed Reflector Attacks [65], and Domain Name System (DNS) Amplification Attacks [61].

3.2.4 DDoS Attack Mitigation

Aspects of DDoS attack on a edge computing server are yet to be explored. However, cloud computing has some similarities with the edge computing server. Therefore, we are going to discuss some strategies of DDoS attack mitigation in cloud in this section.

To successfully conduct a DDoS attack against a hosted server, botnets generate a large number of attack packets and pump those packets to queue. To identify the DDoS attack packets and ensure the QoS of legitimate users, more resources needed to be deployed in the affected server to clone multiple IPSs to carry out the task. To achieve the goal cloning of multiple parallel IPSs is proposed which is depicted in Fig. 3.5. The number of IPSs that will be needed to achieve the goal of mitigating attacks depends on the magnitude of the attack. Botnet's ability to carry out a DDoS attack is usually confined. Therefore, resources needed to beat the attack is not very large. Usually, it is rational to contemplate that a cloud can engage its reserved or inactive resources to fulfill the excess demand. DDoS defense mechanism in cloud ultimately relies on available resources, rather than which particular defense method is used [87].

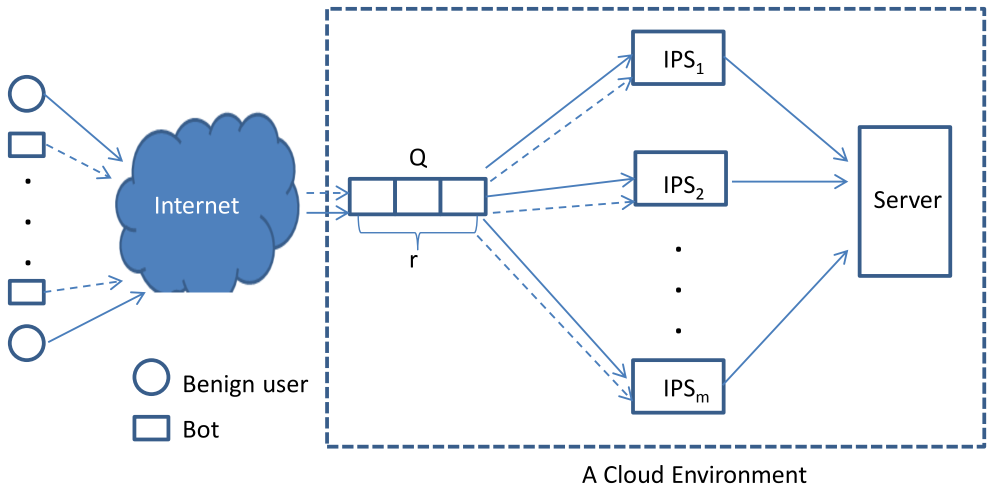


FIGURE 3.5: Cloud server under a DDoS attack with attack mitigation strategy [87]

The attack duration has a huge effect on the services which are running on the cloud. Because of the on-demand nature of the utility computing model in the cloud infrastructure. Economic losses that are incurred due to the DDoS attacks have several components or signs, some of which are easily detectable during the attack. On the other hand, impact of the other losses are visible only after the attack diminishes. It is difficult to measure most of the losses for example, long-term reputations and consequent losses in the business. Exploring DDoS attacks and finding its mitigation solutions is a top research area for the community of cybersecurity. Recently, research on DDoS attacks has seen its focus to shift toward cloud services after the unfolding and espousal of cloud computing. The scale of the DDoS attacks on the cloud is usually volumetric or huge with an attack bandwidths greater than 100 Gbps. However, there are a some DDoS attack occurrences of highly sophisticated or intelligent attacks. In this type of attack the attackers launched a low-rate DDoS attack to beat the attack detection procedure [78].

A DDoS attacks tolerant system is more pragmatic because it is very tough to correctly identify DDoS attacks and prevent them in a timely manner. A DDoS attacks tolerant system is one which is designed including a fault-tolerant technique. Moreover, it can function correctly despite attacks. For example, the fault tolerant system is expected to provide services which meet the specifications of a service level agreement (SLA) even if the system is under an attack by starting automatic operations to revive and recover the affected services and resources. Properties like redundancy, diversity, and independence have to be present in a DDoS attack tolerant system [69].

3.3 Recent DDoS Attack Trend

DDoS attacks are one of the eminent security threats in today's world. There have been numerous DDoS attacks targeting various well known organizations in recent past. In this section we will explore some statistics of the very recent DDoS attacks. In Fig. 3.6 we can see the percentage of number of DDoS attacks in first quarter of 2020 and first quarter and fourth quarter of 2019. The value of first quarter on 2019 is taken as base for data. In quarter four number of attacks decreased than quarter one of 2019. However, in first quarter of 2020 number of attacks increased 180 percent of that of first quarter of 2019. On the right side of the Fig. 3.6 the statistics for smart attacks are shown. Smart attacks are those which are more technically sophisticated and require more ingenuity. For the smart attacks the trend is similar to that of total attacks. Here, the number of smart attacks is 186 percent in first quarter of 2020 compared to that of first quarter of 2019 [63].

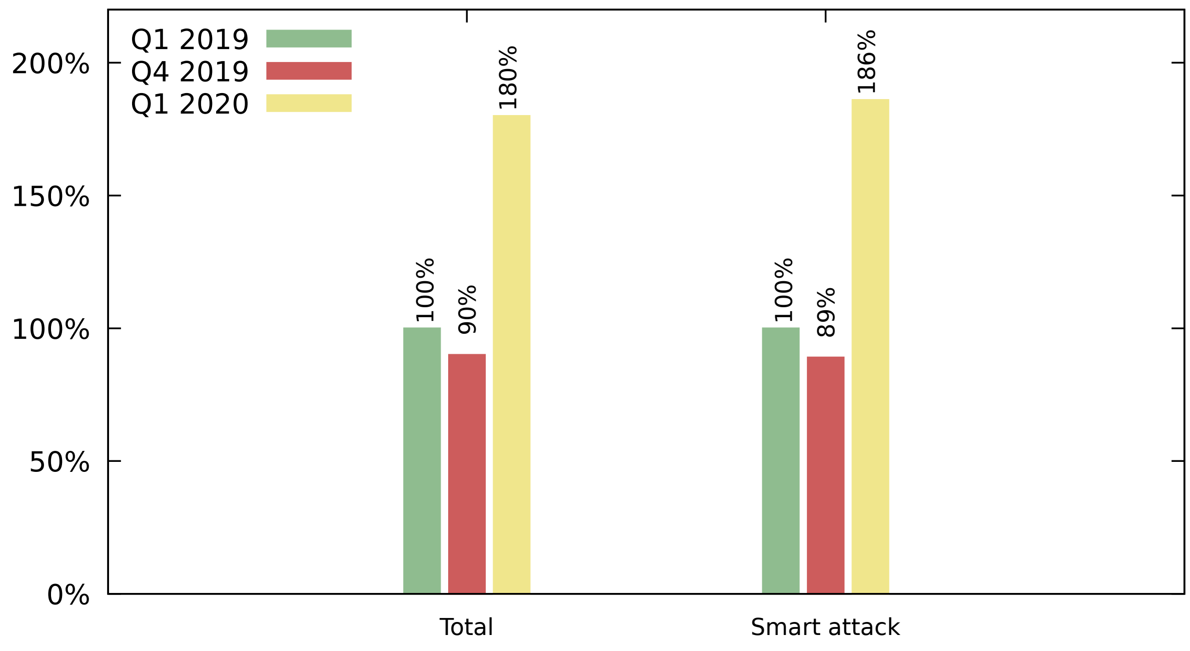


FIGURE 3.6: Percentage of number of DDoS attacks in Q1 2020 and Q1 and Q4 2019 [63]

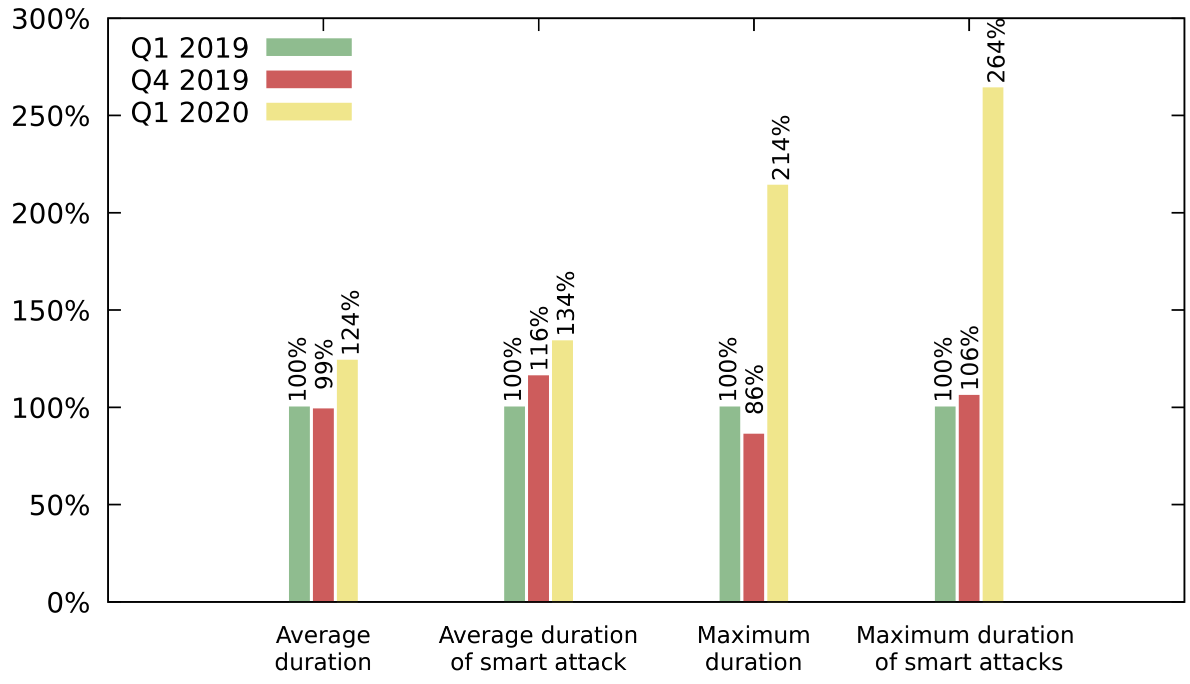


FIGURE 3.7: Duration of DDoS attacks in Q1 2020 and Q1 and Q4 2019 [63]

In Fig. 3.7 data of duration of DDoS attacks in first quarter of 2020 and first quarter and fourth quarter of 2019 are illustrated. Trends a similar for both total attacks and smart attacks. In the first quarter of 2020 average duration for total and smart attacks are 124 percent and 134 percent respectively compared to that of first quarter of 2019. Maximum duration was much larger in the first quarter of 2020 for both total attacks and smart attacks with 214 percent and 264 percent respectively in comparison to that of first quarter of 2019.

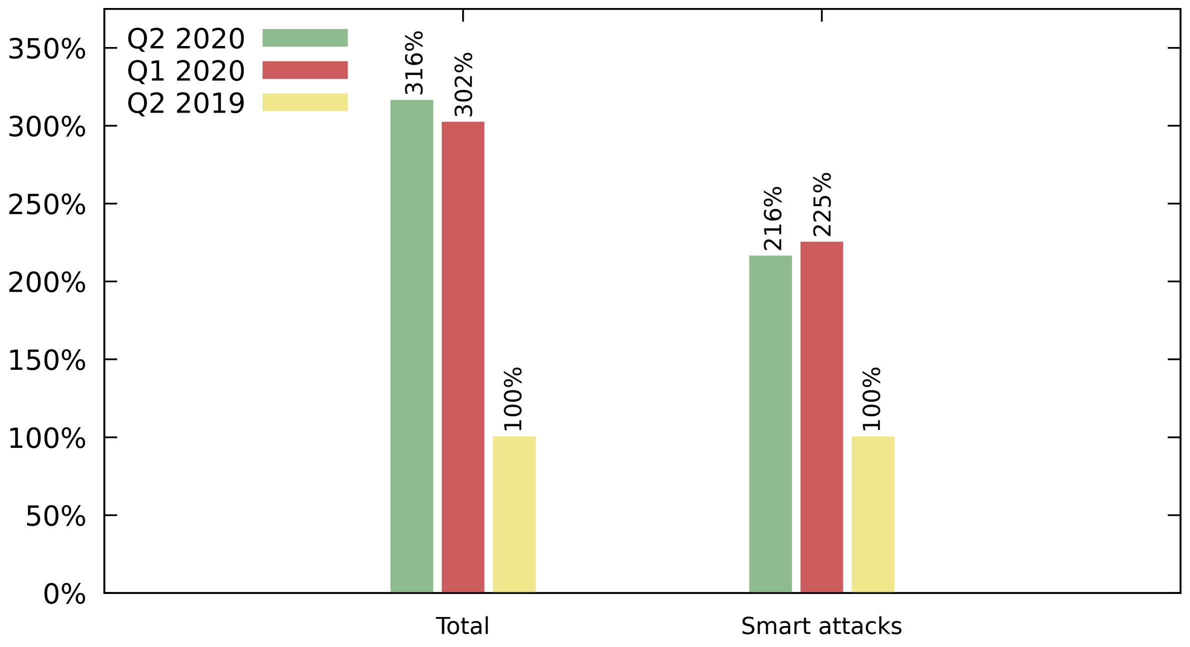


FIGURE 3.8: Percentage of number of DDoS attacks in Q1 and Q2 2020, and Q2 2019 [64]

Fig. 3.8 depicts the percentage of number of DDoS attacks in the first quarter and second quarter of 2020, and second quarter of 2019. Here, total number of attacks in second quarter of 2020 is about 316 percent of that of second quarter of 2019, which can be considered as a huge increase. On the other hand percentage of smart attacks is little less in second quarter of 2020 than that of first quarter of 2020. However, it is still about 216 percent of that of second quarter in 2019 [64].

Average attack duration is shown in Fig. 3.9 for first and second quarters of 2020, and second quarter of 2019. For average duration in total attacks it has decreased in second quarter of 2020 compared to both first quarter of 2020 and second quarter of 2019. For smart attacks average duration has not changed much. However, in total attacks maximum duration in second quarter of 2020 is about 448 percent of that of second quarter of 2019 which is also same for smart attacks [64].

Fig. 3.10 shows the distribution of DDoS attacks by country for first and second quarter of 2020. Highest number attacks were observed in China which is around 61 and 65 percent in first and second quarters respectively. According to number of DDoS attacks the United States is second in the list. USA observed around 19 and 20 percent of attacks in the first and second quarter of 2020. Next country is Hong Kong which observed around 7 and 6 percent in first and second quarter of 2020 respectively. Other countries like Republic of South Africa, Singapore, Australia, India etc. observed very little percentage of DDoS attacks.

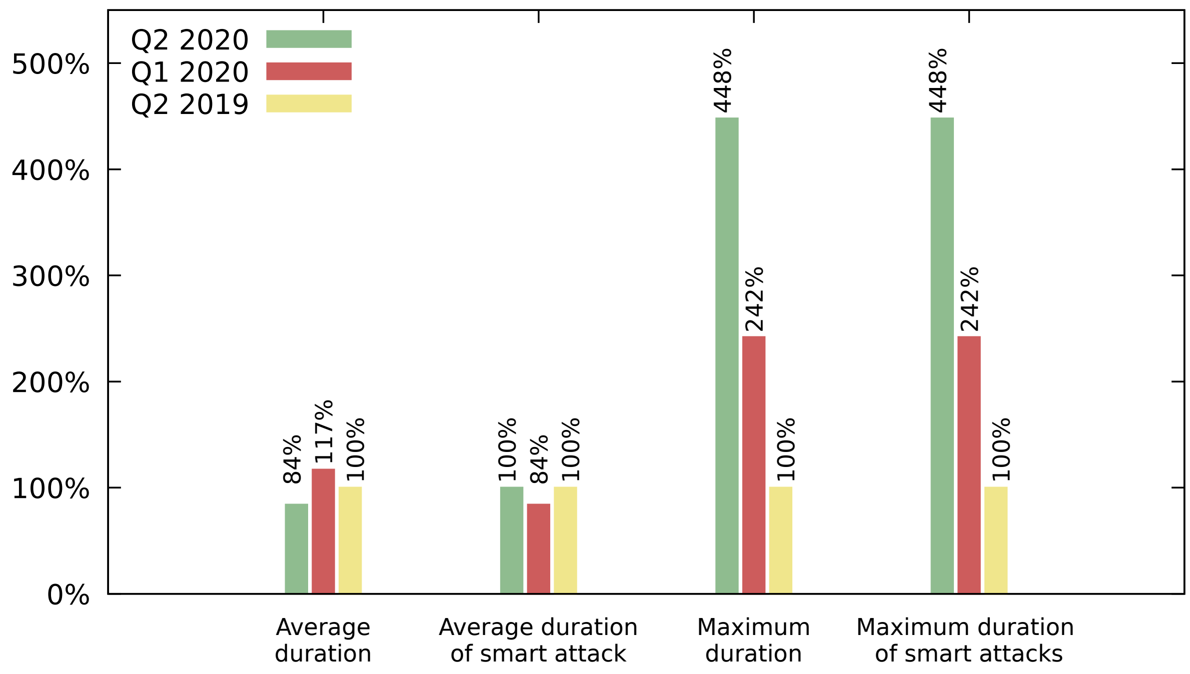


FIGURE 3.9: Duration of DDoS attacks in Q1 and Q2 2020, and Q2 2019 [64]

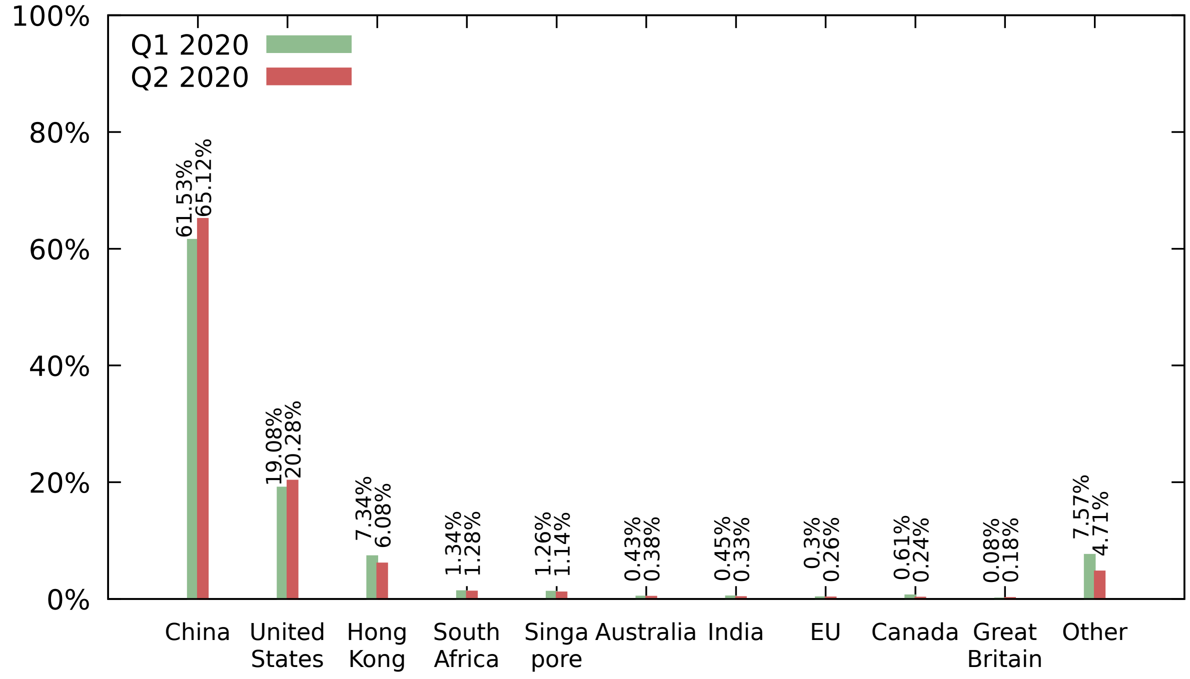


FIGURE 3.10: Distribution of DDoS attacks by country, Q1 and Q2 2020 [64]

Chapter 4

Computation Offloading in Edge Computing

Due to limited capacity of end devices, in edge computing paradigm, tasks are fully or partially offloaded to other neighboring devices or to the edge servers. With the rise in popularity of smartphones, lots of new mobile applications like face recognition, natural language processing, interactive gaming, and augmented reality are coming out and drawing more and more attention. This type of mobile applications is usually resource-hungry, requiring rigorous computation and consuming high energy. Because of the limited the physical size, mobile devices are usually resource-constrained, with constrained computation resources and battery life [18].

In order to overcome these limitations a really promising solution is task offloading which has become a prime focus in academia and also in industry. In the past years, numerous researchers focused on mobile cloud computing in which a mobile users can offload their resource-hungry and computation-intensive tasks to the remote clouds which are resource-rich, via wireless network. This methodology is already employed as a form of commercial cloud services for example Windows Azure, however, it seldom suffers from weak and unreliable wireless connections and delay in wide area network (WAN) between clouds and mobile devices [23]. Fog computing, specifically mobile edge computing is an extension of mobile cloud computing. It is a growing paradigm which utilizes a multitude of collaborative end-user and/or near-user devices to transfer some amount of computation tasks to get the required services with comparatively less delay. As we know the fog computing is implemented at the edge of a network. Therefore, it can offer low-latency services and also agile computation augmenting services devices of end users [9].

As illustrated in Fig. 4.1, authors in [15] propose HyFog. This is an unorthodox hybrid task offloading scheme for fog computing. In this scheme users have the liberty of selecting among several approaches for executing tasks, which includes: (1) Device-to-Device (D2D) Offloaded Execution – devices located in the proximity at the network edge can share the computation resources among each other through task offloading and both can be benefited; (2) Cloud Offloaded Execution – the mighty computing ability allowed by the mobile edge cloud can be engaged to augment the task offloading performance; (3) Local Mobile Execution – a device user can also opt for executing the task locally on his mobile device in order to circumvent any extra overhead due to the process of task offloading. An important attribute of the HyFog framework is that different types of computing resources across multiple devices and the cloud at the network the edge is utilized to allow adaptable task offloading selection options among the users.

Mobile cloud computing is envisaged as a favorable approach to mitigate challenges like meeting demand of resource-hungry applications. To run resource-hungry applications in mobile devices, mobile cloud computing can augment the abilities these devices by providing a mechanism to offload the computation to cloud through wireless access. However, mobile users are expected experience long latency for data exchange with the cloud through the wide area network and this is a visible downside of cloud based mobile cloud computing. Humans are highly sensitive to delay and jitter, therefore, long latency would hurt the interactional response of the the application. Moreover, decreasing the latency or delay is very difficult in wide area network. The cloudlet based mobile cloud computing was proposed as a favorable solution to vanquish this limitation [75]. Instead of depending on a remote cloud, the cloudlet based mobile cloud computing utilizes the actual proximity of user device and cloudlet to decrease the delay by offloading

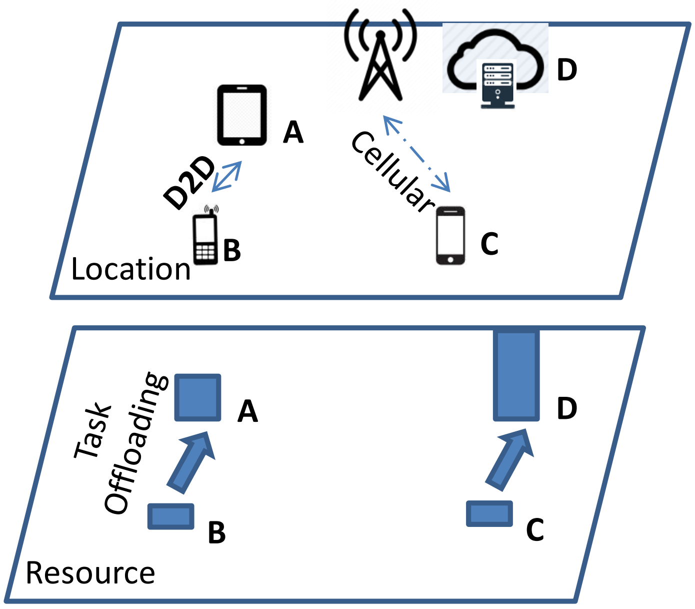


FIGURE 4.1: An example of device-to-device and Cloud offloaded task executions [15]

the computation to the adjacent computing server or cluster through one-hop WiFi wireless access. Unfortunately this cloudlet based scheme has two important drawbacks. Firstly, because of limited coverage of WiFi networks, cloudlet based mobile cloud computing is unable not ensure ubiquitous service provision in all the places. Secondly, because of space limitations, cloudlets usually uses a computing server or cluster with small or medium computation resources. As a result Cloudlets are not able to satisfy QoS of a large number of users [13].

To counter these issues and supplement cloudlet based mobile cloud computing, a unorthodox mobile cloud computing paradigm, called mobile-edge cloud computing, is proposed in [6, 30, 29, 16]. As depicted in Fig. 4.2, cloud-computing capabilities can be provided by mobile-edge cloud computing at the edge of ubiquitous radio access networks in the proximity of the mobile users. In such case, the necessity for fast and interactional response can be satisfied by fast and low-latency connection to the cloud computing infrastructures which are resource-rich installed by telecom operators at the edge of the network. In mobile-edge cloud computing pervasive radio access networks are employed with sturdy computing capabilities. It is envisaged to give pervasive and agile computation supplement services for mobile device users [6, 30, 29, 16].

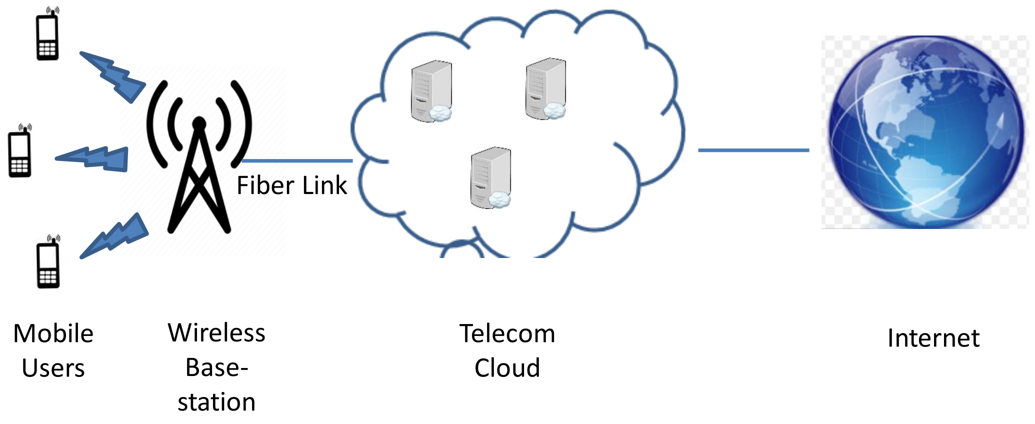


FIGURE 4.2: An example of mobile-edge cloud computing [13]

In the subsections below we will discuss three particular use cases for computation offloading.

4.1 Computation Offloading for Machine Learning Web Apps

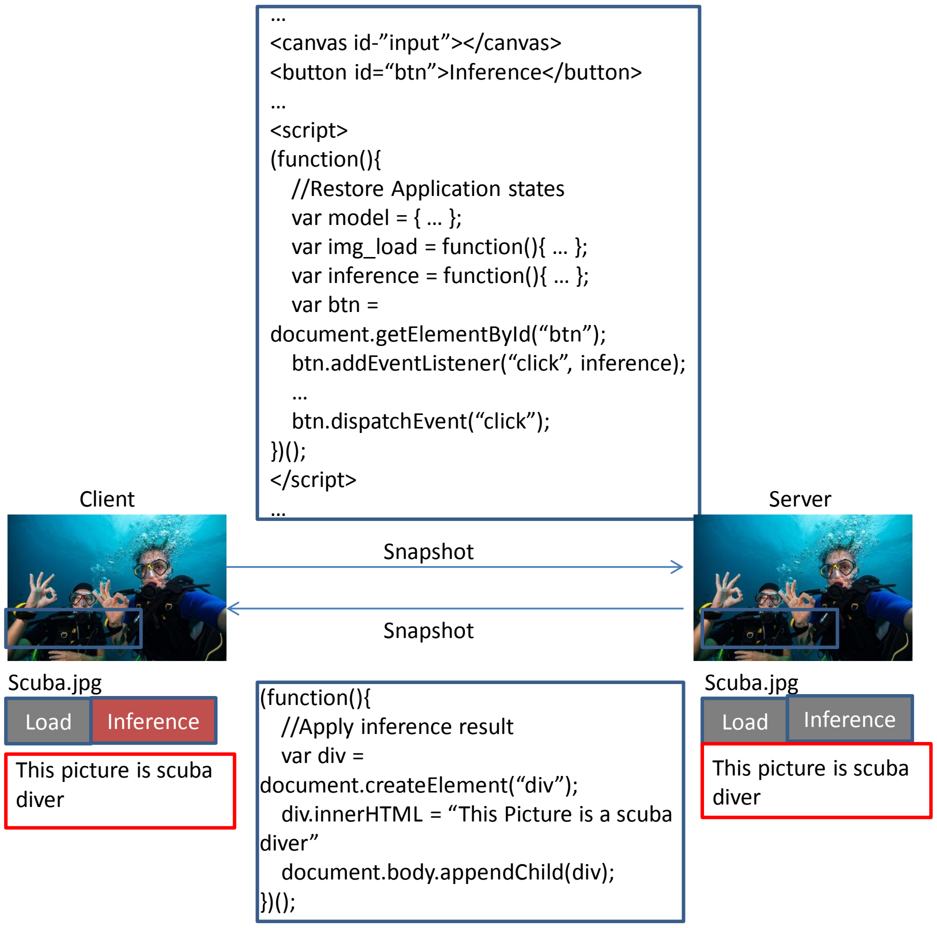
Heavy computations are needed for Machine learning (ML), particularly for training and inference based on a deep neural networks (DNN). This brings in an issue about how we can execute machine learning tasks on embedded devices with limited resources for example wearables or IoT devices. One exciting solution is deploying edge computing [74], which utilizes the concept of decentralized computing infrastructure [10], like fog nodes [9] or cloudlets [75], for offloading required computation.

For ML applications edge servers are considered as specially designed servers for particular applications or as commonly used servers for manifold applications by the current offloading techniques. A computation based server used for video processing is example of such a specially designed server. Here, a streaming video input is sent to the server from a client device. An ML algorithm is applied on the input at the server for a particular application. Finally the server returns the result to the client. Whereas, any computation can be executed at the generic edge servers on demand basis, according to a client's need. To execute this process, in order to customize an edge server ML service software sent from the client. After that this customized server processes the ML requests sent by the client. Using a virtual machine (VM) an edge server can be customized. A VM image is migrated by the client which encloses the ML software and the base system to the server. This software is then installed in the server for later use [75, 41].

In [50], authors propose a different kind of proposition for offloading ML computations to a generic edge server. A regular and self-contained app is created by a programmer for ML which is runnable on the device of the client. The whole app state of the ML is transferred to an adjacent edge server when there is a computation-intensive part ready execution. This app continuously gets executed and after updating the execution state it goes back to the client. A snapshot is captured before starting execution of a computations-intensive part. Then this snapshot is sent from the client to the edge server. The server runs the snapshot on its browser and completing the computation it takes a new snapshot containing the result. The newly captured snapshot is then sent back to client for running on its browser resume its execution. As a result, ML Computations can be offloaded to the edge server on demand basis. Snapshot-based offloading contains the app execution state consisting only HTML/CSS/JavaScript code. Therefore, it is a much lightweight technique in comparison to VM-based strategy as it contains image of the entire disk and memory state. This kind of web apps is runnable in any type of devices and servers with a web browser, therefore, it can be said that snapshot-based offloading is very portable. With respect to specialized ML offloading, the snapshot gives an option for more flexibility in offloading because any type of computations including ML algorithm to be included in the offloading process. The app performance is improved further as it is possible to offload more computations compared to ML specialized edge server [50].

Authors in [50] developed a snapshot-based offloading scheme for ML applications which are DNN-based to be used on Webkit browser. In their proposed system different optimization were made in order to decrease the transmission time of the DNN model. Through these optimizations privacy of user's data was also enhanced. Moreover, authors discussed how offloading system can be installed on-demand basis on the edge servers. In case of real web apps, this snapshot-based offloading scheme demonstrates a favorable performance result, rather than running the app fully on the server, that too considering the snapshot-related overhead.

In the author's proposed scheme the snapshot is transmitted to the server. After that it is run on the server's browser. At first, the execution state present at the time when snapshot was taken by client is restored by the server through executing it. Then server goes on with the execution to process the inference event handler. Later on, when the server completes its processing it captures a new snapshot with the new execution state containing the computation results which are added to the DOM-tree. This is the Javascript code for updating execution state at the client side. This new snapshot is transmitted back to the client. Client then executes this snapshot and goes on with the execution of the app. This is process of computation offloading is depicted in Fig. 4.3 with an example of two snapshots. The offloading process can be carried on smoothly by exchanging the snapshot between the server and the client [50].



```
...
<canvas id-"input"></canvas>
<button id="btn">Inference</button>
...
<script>
(function(){
   //Restore Application states
   var model = { ... };
   var img_load = function(){ ... };
   var inference = function(){ ... };
   var btn =
document.getElementById("btn");
   btn.addEventListener("click", inference);
   ...
   btn.dispatchEvent("click");
})();
</script>
...
```

```
(function(){
   //Apply inference result
   var div =
document.createElement("div");
   div.innerHTML = "This Picture is a scuba
diver"
   document.body.appendChild(div);
})();
```

FIGURE 4.3: An example of snapshot based offloading for a demo app [50]

4.2 Dynamic Computation Offloading for Vehicular Communication System

Computation offloading in vehicular applications is very challenging mainly due to the presence of huge sensory data inputs, stern latency requirements, and dynamic nature of wireless networking situation. Cloud computing is a promising solution to extend vehicular information technology resources. However, execution of vehicular applications remotely is a difficult task for existence of some unique characteristics in the vehicular use cases and their applications. Day by day vehicular applications are becoming more and more reliant on large array of sensory input data. Moreover, large sensory inputs are engaged in interactional applications like those using computer vision an example application is one that includes processing high definition video as shown in Fig. 4.4. Offloading the computations in such applications needs to transfer large sensory data to the cloud. This transfer may need huge time subject to conditions of the network. Applications that are interactive in nature are sensitive to latency. Therefore, a large network round–trip time will ultimately subvert the usefulness speedy resources of cloud computing. It is more challenging to offload computation while the vehicle is in motion. Because in this case network connectivity becomes an issue. Moreover, vehicles may go out of network in some locations [3].

To tackle the issues in offloading vehicular applications authors in [3] propose new system. In this system computation offloading occurs selectively. Rather than offloading the total application only some parts of it is offloaded. According to design concept of the proposed offloading scheme the whole application is separated into modules. Based on the computation requirements the

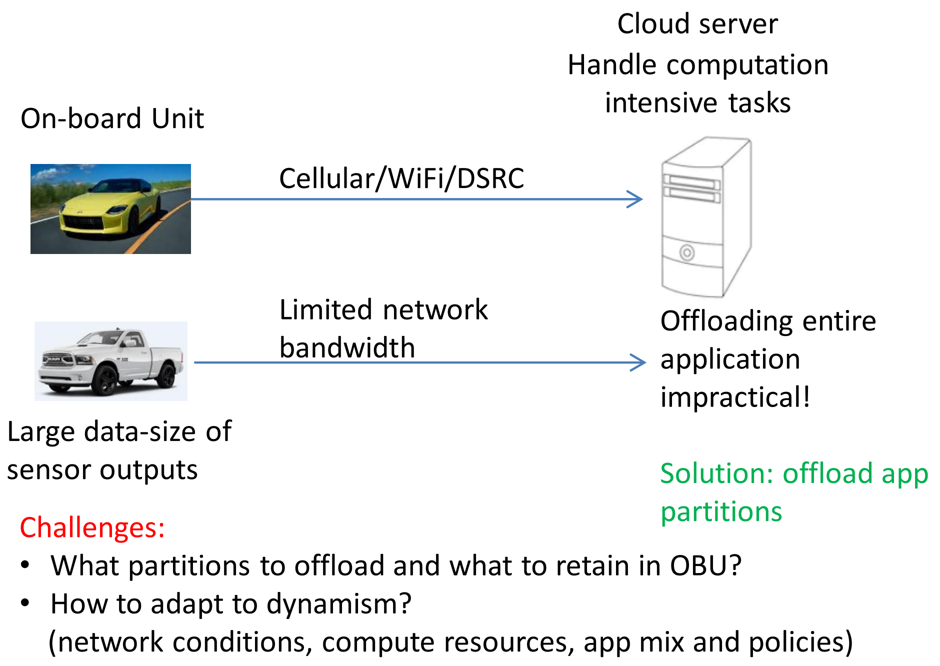

Cloud server
Handle computation intensive tasks
On-board Unit
Cellular/WiFi/DSRC
Limited network bandwidth
Offloading entire application impractical!
Large data-size of sensor outputs
Solution: offload app partitions
Challenges:
• What partitions to offload and what to retain in OBU?
• How to adapt to dynamism?
(network conditions, compute resources, app mix and policies)

FIGURE 4.4: Concept and challenges of offloading in vehicular applications [3]

modules to be offloaded are segregated from the modules which will be executed in the local onboard unit. After separating two types of modules, modules to be offloaded are sent to the cloud for execution. In this proposed design model it is assumed that module definitions are present in the cloud infrastructure.

In the vehicular context, while designing an offloading scheme some key challenges include real-time resource allocation, execution, and adaptation. In order to mitigate these challenges an adaptive cloud-offloading system which includes a heuristic mechanism for dynamic resource allocation is proposed in [3]. The mechanism divides applications into tasks and schedules placements of these tasks to be executed in Onboard Unit (OBU) or cloud depending on variations in network bandwidth, OBU processor cycles availability, and policies. Thus it is the objective of the mechanism to identify when a schedule is feasible. This is done before applications are ready to be scheduled at run–time and the schedule can adapt accordingly on the basis of the planning made by resource allocation. The objective of this offloading scheme design is guaranteeing that the response time of the vehicular applications are limited within its specified deadlines. Here, "response time" is defined as the total time needed for execution of the application. This execution time is measured from initiation to completion of the computation. and "deadline" is defined as the maximum time duration within which the application needs to finish its execution in order to be considered as useful by the user.

4.3 Computation Offloading from IoT devices

In [56] authors proposed a selective offloading scheme. This scheme is designed to reduce consumption of energy of the devices. In this scheme the devices are designed to be self-nominated or self-denied for offloading in order to decrease the signaling overhead. Several computation-intensive services, including e-health, automatic driving, and industrial automation, are developing and surpassing the computing and storage abilities of IoT devices. Cloud computing is designed to possess huge storage and powerful computing facilities. On the other hand IoT devices have limited storage and computing capabilities. Therefore, IoT devices can offload storage and computations to cloud in order to elongate their lifetime [71]. However, due to long distance from IoT devices cloud can not be considered is suitable offloading option for delay-sensitive operations. Therefore, MEC can be used to perform such operations. However, limited resource capacity is a bottleneck for edge servers. As a result, scalability is a key problem in MEC [85, 86, 14, 53]. This brings in a trade-off between the scale of offloading and the quality of service

(QoS). The existing literature solves the scalability issue by either engaging the load balancing in the edge servers to accumulate and endure the workloads [85, 86] or managing the coordination among mobile devices to select services for offloading [14, 53].

Authors enclose the requirements of the latency determined at particular devices in their offloading requests in order to eliminate the dependency of task partitioning of different devices. The proposed request and admission framework in [56] solves the inherent scalability problem of MEC. This framework can be operated at the devices and edge servers individually, without requiring any coordination among devices. With an objective to reduce energy consumption of the IoT devices authors propose a selective offloading scheme with the help of request and admission framework. Through this framework latency requirements of different services are also satisfied [56].

The working process of the framework mainly contains firstly, creating the offloading requests at the particular devices. Secondly, reserving the resources for the latency specifications at the resource scheduler. Thirdly, utilizing the diversity among devices for choosing the energy-saving services in case of offloading at the admission controller.

The delay-sensitive tasks are given higher preference for processing in the proposed scheme. On the other hand the delay-tolerant tasks are kept in queue at the edge server under heavy workload. Therefore, delay-tolerant tasks are executed at the edge servers only when the workload is light. Sometimes the queued tasks are offloaded to cloud in order to avoid excessive delay. Below the steps of selective offloading scheme are discussed. Requests are independently made at the mobile devices.

A usual remote execution has three stages:

1. Uploading the input
2. Remote execution at the edge server
3. Receiving the computation result.

The synchronizer accepts the requests for offloading at the server end. Then the problem that has to be considered is choosing and allocating resources to the offloaded tasks while keeping the energy consumption to a minimum and also satisfying the latency specifications of all the pending offloading requests. It is important that tasks are executed meeting the deadlines. The diversity of devices and their IoT services makes offloading more appropriate for some of the devices, and execution at the device itself more suitable for other devices [56].

4.4 Relation Between Computation Offloading and DDoS Attacks

In previous sections we discussed computation offloading for three use cases. These computations are offloaded depending upon particular situation. For example, when a computation cannot be executed in a device for resource limitation it is offloaded to edge servers. This is the perspective of offloading from device to server. Moreover, offloading may take place from one edge server to another. In both the cases workload increases in the edge server. In this way an edge server may become overloaded. Moreover, an edge server tries to offload some of its tasks to another edge server near to it when it becomes overloaded. Therefore, edge servers can become sequentially overloaded through this kind of offloading. DDoS attacks can exploit this scenario. The basic aspect of DDoS attacks is to send huge number of packets to a particular server to exhaust its resources. Therefore, when an edge server is targeted for DDoS attack, the attacker will send lots of data packets to it. Initially the edge server will try to cope up with the situation by offloading its tasks to a nearby edge server. However, eventually the affected edge server will be exhausted and also there will be possibility that some of its nearby edge servers also become exhausted. Therefore, there is a close relationship between task offloading and DDoS attacks.

4.5 Recovery Schemes in an Overloaded Edge Server

In this section we will discuss the recovery schemes of MEC servers when they are overloaded. MEC provides an opportunity for different services like augmented reality, virtual reality, etc. to operate with low latency, location awareness and mobility support in order to mitigate disadvantages of cloud computing. However, these MECs can be overloaded by tasks due to high volume of tasks request. Moreover, a MEC can be out of service due to failure. Therefore, quality of experience (QoE) and advantages of MEC will be diminished.

Authors in [73] proposed two recovery schemes for an overloaded or failed MEC. According to recovery scheme I the Overloaded-MEC (O-MEC) is enabled to offload its tasks to other MECs which are inside the same cluster. When there is no nearby MEC in proximity of O-MEC, it will go with activating the recovery scheme II for offloading its tasks to neighboring MECs through ad-hoc relay nodes (RNs). If a MEC goes out of service, MECs which are in the same cluster can help it by allocating their resources. Fig. 4.5 depicts the first recovery scheme. In Fig. 4.5, demonstrates that MEC 1 has failed due to being overwhelmed by requests of tasks. MEC1 updates the cluster head, MEC3 about its situation. MEC3 looks for an MEC which is present in the range of wireless radio in the same cluster as that of MEC1 to help it. Then MEC3 notifies MEC1 about the location of the MEC which is available for task offloading. After that, MEC1 transfers the extra workload directly to the other MECs which are available, in order to recover from the service failure.

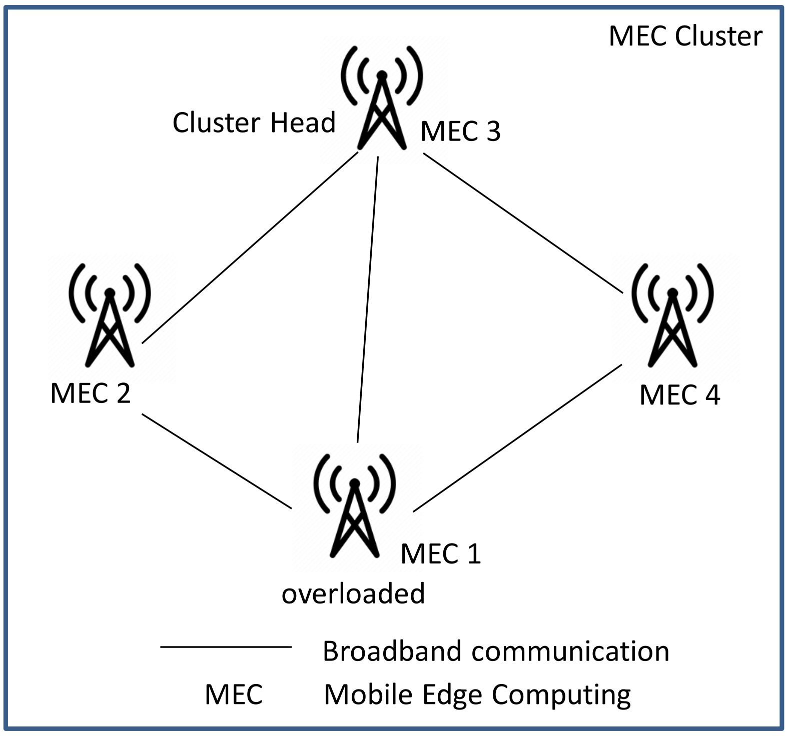


FIGURE 4.5: Offloading in the MEC cluster according to recovery scheme I [73]

It may happen that there is no available MEC within the cluster of O-MEC or all the MECs are performing their designated tasks with full capacity as illustrated in Fig. 4.6. Therefore, O-MEC cannot provide its designated services. To tackle such situation, authors present a solution in which the user devices of an available MEC present within cluster of the O-MEC become ad-hoc RNs. Here, the ad-hoc RNs are not any base station, they are just users in adjacent MECs. Fig. 4.6 shows that the O-MEC is overloaded. As a result, the users' devices in the cluster of O-MEC are disconnected and cannot get any service from that MEC. There are two kinds of disconnected user devices: an "adjacent disconnected node" is one which is adjacent to an ad-hoc relay node and a "normal disconnected node" which does not have any adjacent ad-hoc relay node. Cluster head of disconnected user devices transmits a query to its adjacent ad-hoc RNs in order to discover if they are currently connected to any of the available recovery MEC. If there exists such a neighboring ad-hoc RNs which are connected to an available recovery MEC, then the disconnected user devices residing within the cluster of the O-MEC will look for possible paths among all pairs of devices using the Floyd–Warshall algorithm. The disconnected user devices within the range of the O-MEC are categorized into two parts as shown in Fig. 4.6. On the left side, there are user

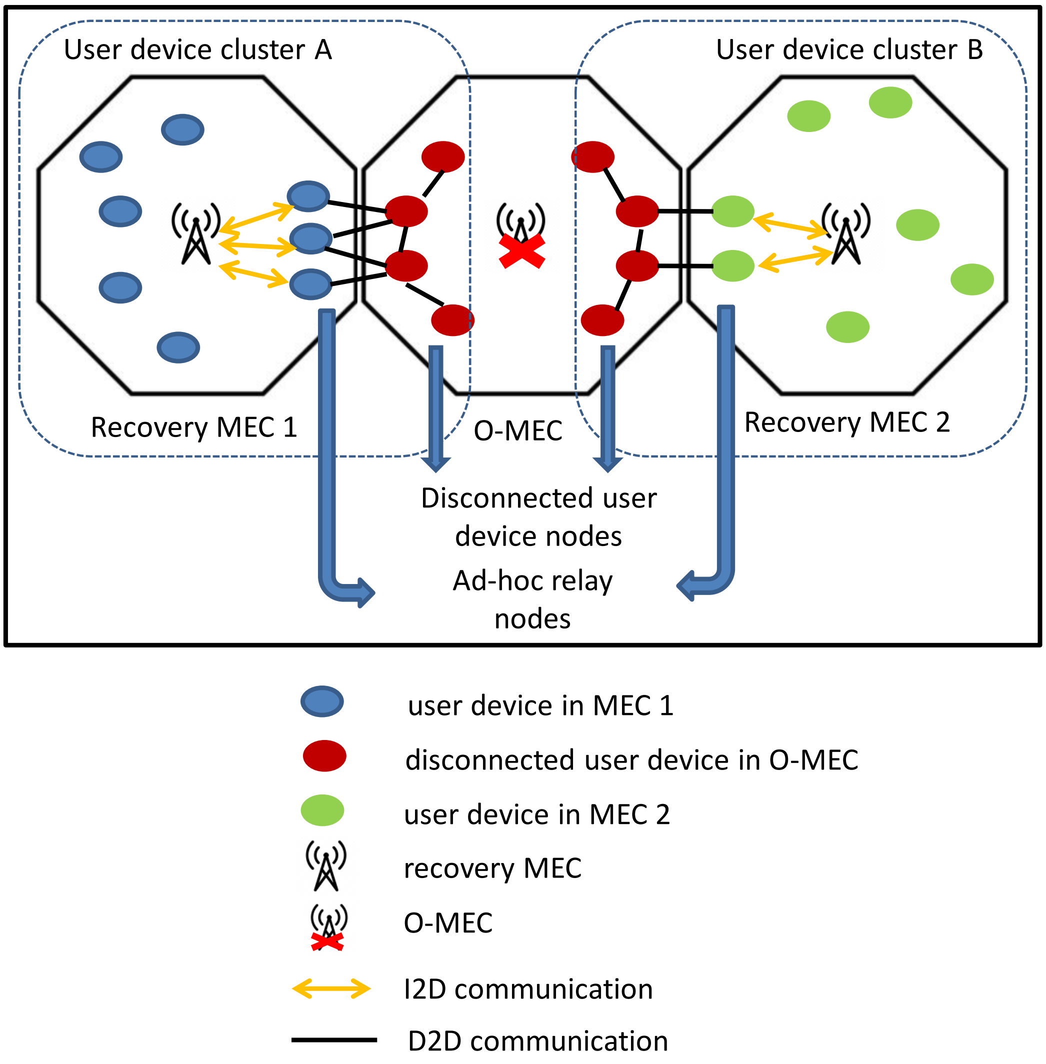


FIGURE 4.6: Recovery scheme II [73]

devices in the cluster A developing a multihop wireless network, on the other hand, on the right side, there are user devices in the cluster B forming another multihop wireless network. Fig. 4.7 depicts a multihop wireless network needed for the recovery of disconnected user devices inside the cluster of the O-MEC. According to the second recovery scheme, the recovery MEC accepts service requests from the disconnected user devices through the ad-hoc RNs. Within the cluster of recovery MEC, the role of ad-hoc RN at the border of the O-MEC is played by a user device. In this way, in recovery scheme II the user devices of O-MEC gets the required service through RNs [73].

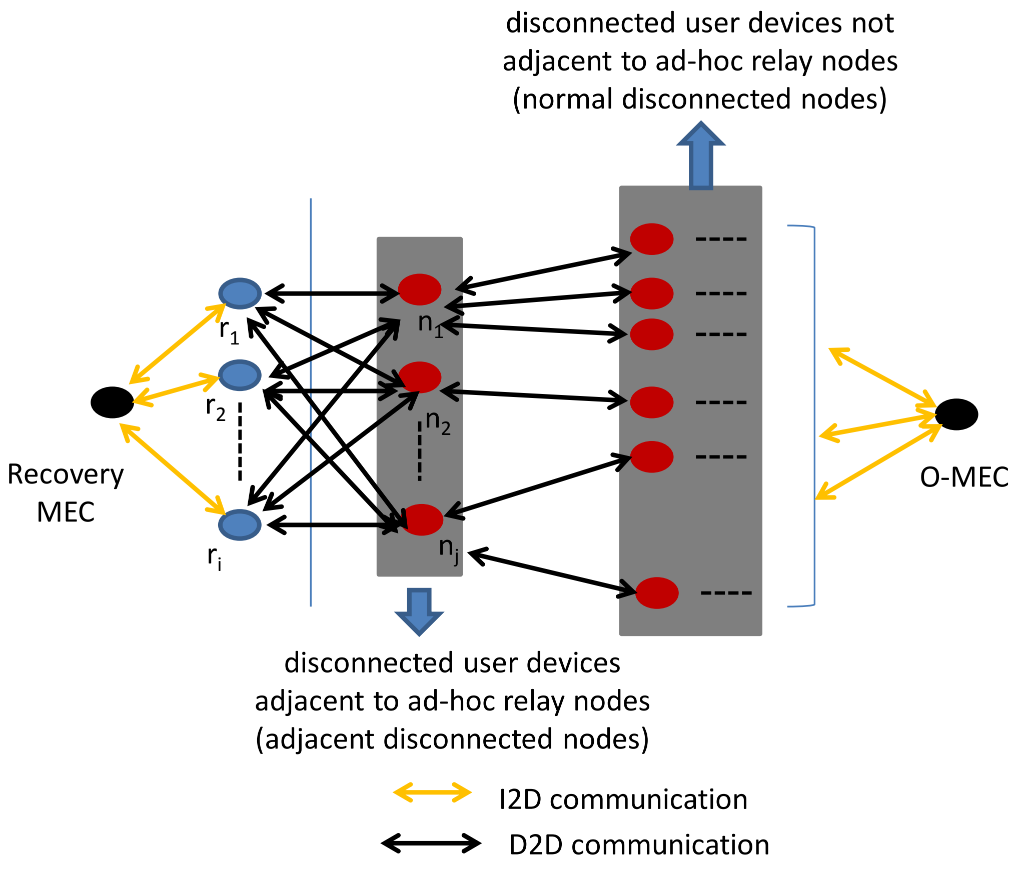

disconnected user devices not adjacent to ad-hoc relay nodes (normal disconnected nodes)
r1
r2
ri
n1
n2
nj
Recovery MEC
O-MEC
disconnected user devices adjacent to ad-hoc relay nodes (adjacent disconnected nodes)
I2D communication
D2D communication

FIGURE 4.7: Multihop wireless network along with ad-hoc RNs and disconnected nodes according to recovery scheme II [73]

Chapter 5

How Attack Can Be Launched in Edge Servers

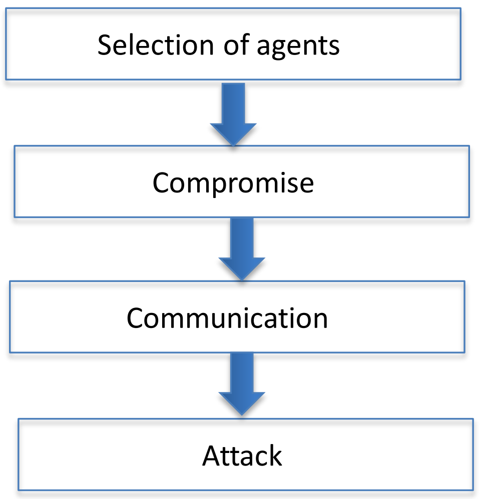


FIGURE 5.1: Generic steps to perform a DDOS attack

The generic steps to perform a DDOS attack to a server are shown in Fig. 5.1. The very first step is to select the agents. Agents are available all over the internet. In the first step of launching the DDoS attack some agents are selected from the available agents. In the second step number of devices available in the internet are compromised for launching the attack. In this step security vulnerabilities of the available devices are exploited to compromise them. This compromisation are done by the attacker with the help of selected agents. The next step is to communicate. In this step the attacker communicates with the compromised devices. The attacker can perform this communication either directly or through agents. The final step is attack. All the necessary prerequisites are completed through initial three steps. In the final step the attacker commands all the compromised devices to launch attack to a particular target server or the victim.

Now the details of the attack will be discussed. DDoS attacks can be carried out following different methodologies. However, to attack an edge server two common strategies can be deployed. These strategies are:

1. Using reflection method
2. Using botnets

Now these two strategies will be discussed in brief. The first one is reflection method. In this method the attacker assumes the identity of the victim by forging its IP address. With this spoofed identity attacker will send out countless Domain Name System (DNS) queries to an open DNS resolver. The DNS resolver processes each query and then sends the information to the victim edge server as its IP address was spoofed in the incoming DNS queries. These information packets sent by the DNS resolver are much bigger than the queries the DNS resolver receives. This reflection method is depicted in Fig, 5.2. Here, the attacker with IP address for example 198.51.100.2 sends out DNS queries to the reflector with source IP address 198.51.100.3 (the IP address of victim edge server). After processing the queries the reflector will send a bulk amount of response data packets to the victim edge server. Due to this bulk amount of data packets flooded to it, the edge server will fail to process all the data packets and eventually get exhausted. This exhausted edge server will no longer be available for performing its usual operations due to being victim of a DDoS attack.

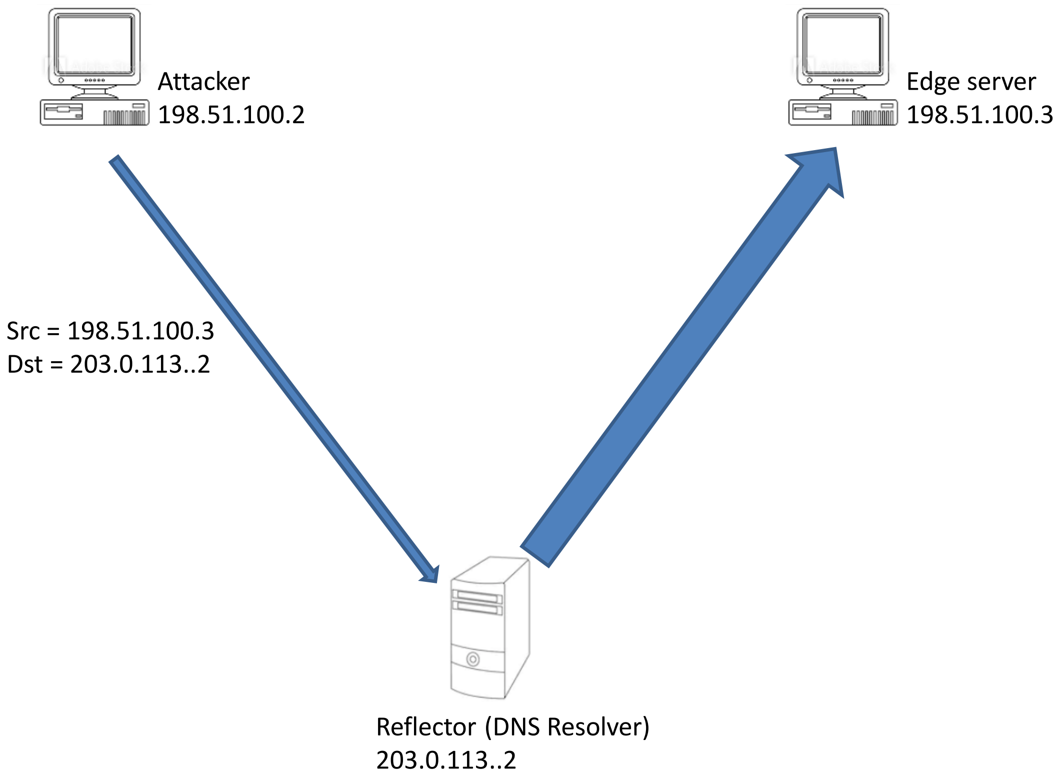


FIGURE 5.2: Reflection method of DDOS attack

Now DDoS attack through botnet strategy will be discussed. In this method firstly an edge server or target will be selected to launch a DDoS attack. Then a large amount of botnets are managed. However, compared to traditional DDoS attacks, the number of botnets needed to attack an edge server will be less because the capacity of edge server is limited. The botnets can be managed in two ways. Either existing benign botnets are compromised or botnets can be hired to launch a DDoS attack. After managing required amount of botnets they are deployed and attack is launched to exhaust the target edge server.

Chapter 6

Summary

In this section we present a summary table of our survey. Table 6.1 shows the major state-of-art literatures surveyed in this article. Main research area covered in this survey are edge computing, various aspects of DDoS attack, computation offloading in edge servers and how a real life attack can be launched targeting an edge server. The state-of-art literatures representing research in these areas are mentioned in the the summary table 6.1. This table well summarizes the survey conducted in this research article.

Summary Table			
Source	Focus	Findings	Impact
Kashif et al. [8]	Definition, motivation, advantages of edge paradigms	Challenges and use-cases of edge paradigms	Guides to trends and future opportunities of edge paradigms
Rodrigo et al. [72]	Security threats, challenges of edge paradigms	Impacts of security attacks on edge paradigms	Guides to encounter and mitigate security issues
Tom et al. [55]	Concept, architecture and design goals of Fog	Purpose and use-case of Fog	Improvement in user service quality and bandwidth consumption by Fog
Dolui et al. [26]	Implementations of Fog, Cloudlet and MEC	Standardization of Fog, Cloudlet and MEC	Made a decision tree for selecting among edge implementations
Yun et al. [45]	Service and deployment scenarios of MEC	How proximity, network and context information utilized in MEC	Established Proof-of-Concept to demonstrate viability of MEC
Esraa et al. [2]	Botnet based DDoS attacks	Pros and cons of botnet based attacks	Eases future study on attacks
Christos et al. [27]	Structure of DDoS attacks and defense mechanism	Pros and Cons of attack and defense categories	Facilitates better understanding of attacks and powerful algorithms for defense
Yu et al. [87]	Dynamic resource allocation method to defend DDoS attacks	Use of multiple intrusion prevention servers to filter out attack packets	Mathematical model to prove DDoS attacks can be defeated
Krishna et al. [69]	Relationship between SDN and DDoS attacks	Challanges to mitigate DDoS attacks in SDN	How advantages of SDN counter DDoS attacks
Felix et al. [36]	DDoS attack prevention methods	Root-cause of DDoS attacks	Shuts down control network of IRC-based DDoS attacks
Somani et al. [78]	DDoS attacks, their variants and attack dynamics	Attack methods, consequences and lessons from past attacks	Multilevel alert flow-based collaborative DDoS detection framework
Jeong et al. [50]	Machine learning based task offloading scheme	Efficient snapshot based offloading method	Introduces app-level customization for edge server
Ashwin et al. [3]	Vehicular data offloading	How offloading works in vehicular applications	Heuristic mechanisms for partitioning applications into modules
Lyu et al. [56]	Selective offloading scheme for IoT devices	How IoT device can do computation intensive tasks	Minimize energy consumption and signaling overhead of IoT devices
Dimas et al. [73]	Recovery strategies for overloaded edge servers	Neighboring MECs can mitigate the problem of overloaded MEC	Provision for overloaded MEC to recover

TABLE 6.1: Summary table for the survey on DDoS attacks in edge servers

Chapter 7

Conclusion

This survey was carried out by extensively reviewing state-of-art literatures. Several literatures were surveyed in the domain of edge computing and DDoS attacks and task offloading in edge computing. Therefore, this survey will guide future researchers to practically launch DDoS attacks on edge servers. Moreover, this survey will also help the researchers to propose real life defense mechanism against DDoS attacks in edge servers.

Like other research articles this survey also has some limitations. One major limitation of this survey is that no experiment could be conducted. Another major limitation is that real life results of DDoS attacks on an edge server could not be analyzed in this survey. Also no real life defense mechanism on the edge server against a DDoS attack could be proposed in this survey.

Edge computing is a comparatively new paradigm in the research arena. There are lots of aspect in edge computing which are yet to be completely explored in depth. Edge computing paradigm comes with edge servers with limited capacity to support sophisticated latency-sensitive computations. Therefore, availability of these servers is very important at the time of task-request arrival. However, like other technologies present in the internet, these servers are prone to security attacks. Moreover, impact of such attacks is more disastrous for edge servers as they have limited computation capabilities. Along with different edge computing paradigms, various aspects of DDoS attacks and their impacts on edge servers are discussed in this article. Moreover, various recovery schemes for an overloaded (due to DDoS attack) edge server are depicted in this survey paper. This paper will pave the way for future researchers who want to explore DDoS attack kind of security vulnerabilities in edge servers.

Bibliography

[1] Mufajjul Ali. 2009. Green cloud on the horizon. In *Cloud Computing*. Martin Gilje Jaatun, Gansen Zhao, and Chunming Rong, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 451–459.

[2] Esraa Alomari, Selvakumar Manickam, B. B. Gupta, Shankar Karuppayah, and Rafeef Alfaris. 2012. Botnet-based distributed denial of service (ddos) attacks on web servers: classification and art. *International Journal of Computer Applications*, 49, 7, 24–32. ISSN: 0975-8887. DOI: 10.5120/7640-0724. http://dx.doi.org/10.5120/7640-0724.

[3] Ashwin Ashok, Peter Steenkiste, and Fan Bai. 2018. Vehicular cloud computing through dynamic computation offloading. *Computer Communications*, 120, 125 –137. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2017.12.011. http://www.sciencedirect.com/science/article/pii/S0140366417304140.

[4] Paul Bacher, Thorsten Holz, Markus Kotter, and Georg Wicherski. 2005. Know your enemy: tracking botnets. *The Honeynet Project & Research Alliance*.

[5] Paramvir Bahl, Richard Y. Han, Li Erran Li, and Mahadev Satyanarayanan. 2012. Advancing the state of mobile cloud computing. In *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services* (MCS '12). Association for Computing Machinery, Low Wood Bay, Lake District, UK, 21–28. ISBN: 9781450313193. DOI: 10.1145/2307849.2307856. https://doi.org/10.1145/2307849.2307856.

[6] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo. 2013. Joint allocation of computation and communication resources in multiuser mobile cloud computing. In *2013 IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 26–30.

[7] M. T. Beck and Marco Maier. 2014. Mobile edge computing: challenges for future virtual network embedding algorithms. In.

[8] Kashif Bilal, Osman Khalid, Aiman Erbad, and Samee U. Khan. 2018. Potentials, trends, and prospects in edge technologies: fog, cloudlet, mobile edge, and micro data centers. *Computer Networks*, 130, 94 –120. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2017.10.002. http://www.sciencedirect.com/science/article/pii/S1389128617303778.

[9] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 13–16.

[10] Eugen Borcoci. 2016. Fog computing, mobile edge computing, cloudlets-which one. In *SoftNet Conference*, 1–122.

[11] Simon Byers, Aviel D. Rubin, and David Kormann. 2004. Defending against an internet-based attack on the physical world. *ACM Trans. Internet Technol.*, 4, 3, (August 2004), 239–254. ISSN: 1533-5399. DOI: 10.1145/1013202.1013203. https://doi.org/10.1145/1013202.1013203.

[12] Bysin. 2001. Knight.c sourcecode, *2001. Available at: http://packetstormsecurity.org/distributed/knight.c*.

[13] X. Chen, L. Jiao, W. Li, and X. Fu. 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24, 5, 2795–2808.

[14] X. Chen, L. Jiao, W. Li, and X. Fu. 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24, 5, 2795–2808.

[15] X. Chen and J. Zhang. 2017. When d2d meets cloud: hybrid mobile task offloadings in fog computing. In *2017 IEEE International Conference on Communications (ICC)*, 1–6.

[16] ETSI Mobile Edge Computing, I Initiative, et al. 2014. Mobile-edge computing: introductory technical white paper. *ETSI: Sophia Antipolis, France*, 1–36.

[17] PJ Criscuolo. 2000. Distributed Denial of Service Tools, Trin00, Tribe Flood Network, Tribe Flood Network 2000 and Stacheldraht. Technical report. Lawrence Livermore National Lab., CA (US).

[18] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. 2010. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services* (MobiSys '10). Association for Computing Machinery, San Francisco, California, USA, 49–62. ISBN: 9781605589855. DOI: 10.1145/1814433.1814441. https://doi.org/10.1145/1814433.1814441.

[19] A. V. Dastjerdi and R. Buyya. 2016. Fog computing: helping the internet of things realize its potential. *Computer*, 49, 8, 112–116.

[20] A.V. Dastjerdi, H. Gupta, R.N. Calheiros, S.K. Ghosh, and R. Buyya. 2016. Chapter 4 - fog computing: principles, architectures, and applications. In *Internet of Things*. Rajkumar Buyya and Amir [Vahid Dastjerdi], editors. Morgan Kaufmann, 61 –75. ISBN: 978-0-12-805395-9. DOI: https://doi.org/10.1016/B978-0-12-805395-9.00004-6. http://www.sciencedirect.com/science/article/pii/B9780128053959000046.

[21] A. Davis, J. Parikh, and W. E. Weihl. 2004. Edgecomputing: extending enterprise applications to the edge of the internet. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers Posters* (WWW Alt. '04). Association for Computing Machinery, New York, NY, USA, 180–187. ISBN: 1581139128. DOI: 10.1145/1013367.1013397. https://doi.org/10.1145/1013367.1013397.

[22] Sven Dietrich, Neil Long, and David Dittrich. 2000. Analyzing distributed denial of service tools: the shaft case. In *LISA*, 329–339.

[23] Hoang T Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. 2013. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, 13, 18, 1587–1611.

[24] David Dittrich. 1999. The tribe flood network. *distributed denial of service attack tool, 199l. Available at: http://staff. washington. edu/dittrich/misc/tfn. analysis*.

[25] David Dittrich. 1999. The 'stacheldraht'distributed denial of service attack tool. (1999).

[26] K. Dolui and S. K. Datta. 2017. Comparison of edge computing implementations: fog computing, cloudlet and mobile edge computing. In *2017 Global Internet of Things Summit (GIoTS)*, 1–6.

[27] Christos Douligeris and Aikaterini Mitrokotsa. 2004. Ddos attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44, 5, 643 –666. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2003.10.003. http://www.sciencedirect.com/science/article/pii/S1389128603004250.

[28] Christos Douligeris and Aikaterini Mitrokotsa. 2004. Ddos attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44, 5, 643 –666. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2003.10.003. http://www.sciencedirect.com/science/article/pii/S1389128603004250.

[29] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, and P. Narasimhan. 2013. The case for mobile edge-clouds. In *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, 209–215.

[30] Ericsson. 2012. The telecom cloud opportunity, *March 2012. [Online]. Available: http://www.ericsson.com/res/sit AU/docs/2012/ericsson telecom cloud discussion paper*.

[31] MECISG ETSI. 2014. Mobile edge computing-introductory technical white paper. *etsi2014mobile, no. Issue*. http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing.

[32] MARJZ Fabian and Monrose Andreas Terzis. 2007. My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In *Proceedings of the 1st USENIX Workshop on Hot Topics in Understanding Botnets, Cambridge, USA*, 18.

[33] Jose Oscar Fajardo, Ianire Taboada, and Fidel Liberal. 2015. Radio-aware service-level scheduling to minimize downlink traffic delay through mobile edge computing. In *Mobile Networks and Management*. Ramón Agüero, Thomas Zinner, Mario García-Lozano, Bernd-Ludwig Wenning, and Andreas Timm-Giel, editors. Springer International Publishing, Cham, 121–134.

[34] Farhan Bashir Shaikh and S. Haider. 2011. Security threats in cloud computing. In *2011 International Conference for Internet Technology and Secured Transactions*, 214–219.

[35] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. 2009. Above the clouds: a berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28, 13, 2009.

[36] Felix C. Freiling, Thorsten Holz, and Georg Wicherski. 2005. Botnet tracking: exploring a root-cause methodology to prevent distributed denial-of-service attacks. In *Computer Security – ESORICS 2005*. Sabrina de Capitani di Vimercati, Paul Syverson, and Dieter Gollmann, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 319–335. ISBN: 978-3-540-31981-8.

[37] Neil Long G. W. David Dittrich Sven Dietrich. 2000. The mstream. *distributed denial of service attack tool, 2000. Available at: http://staff.washington.edu/dittrich/misc/mstream.analysis.txt*.

[38] D. Gautam and V. Tokekar. 2017. An approach to analyze the impact of ddos attack on mobile cloud computing. In *2017 International Conference on Information, Communication, Instrumentation and Control (ICICIC)*, 1–6.

[39] B. B. Gupta, R. C. Joshi, and Manoj Misra. 2012. Distributed denial of service prevention techniques. (2012). arXiv: 1208.3557 [cs.CR].

[40] B. B. Gupta, R. C. Joshi, and Manoj Misra. 2012. Dynamic and auto responsive solution for distributed denial-of-service attacks detection in isp network. (2012). arXiv: 1204.5592 [cs.CR].

[41] Kiryong Ha, Padmanabhan Pillai, Wolfgang Richter, Yoshihisa Abe, and Mahadev Satyanarayanan. 2013. Just-in-time provisioning for cyber foraging. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services* (MobiSys '13). Association for Computing Machinery, Taipei, Taiwan, 153–166. ISBN: 9781450316729. DOI: 10.1145/2462456.2464451. https://doi.org/10.1145/2462456.2464451.

[42] N. Hachem, Y. Ben Mustapha, G. G. Granadillo, and H. Debar. 2011. Botnets: lifecycle and taxonomy. In *2011 Conference on Network and Information Systems Security*, 1–8.

[43] Bill Hancock. 2000. Trinity v3, a ddos tool, hits the streets. *Computers & Security*, 19, 7, 574–574.

[44] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita. 2015. Botnet in ddos attacks: trends and challenges. *IEEE Communications Surveys Tutorials*, 17, 4, 2242–2270.

[45] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11, 11, 1–16.

[46] 2013. Ibm news releases, ibm and nokia siemens networks announce world first mobile edge computing platform. http://www-03.ibm.com/press/us/en/pressrelease/40490.wss.

[47] et al J. Rosenberg. 2002. Rfc 3261 sip: session initiation protocol, *2002. Available at: www.ietf.org*.

[48] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa. 2016. The future of mobile cloud computing: integrating cloudlets and mobile edge computing. In *2016 23rd International Conference on Telecommunications (ICT)*, 1–5.

[49] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa. 2016. The future of mobile cloud computing: integrating cloudlets and mobile edge computing. In *2016 23rd International Conference on Telecommunications (ICT)*, 1–5.

[50] H. Jeong, I. Jeong, H. Lee, and S. Moon. 2018. Computation offloading for machine learning web apps in the edge server environment. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 1492–1499. DOI: 10.1109/ICDCS.2018.00154.

[51] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam. 2014. A taxonomy of botnet behavior, detection, and defense. *IEEE Communications Surveys Tutorials*, 16, 2, 898–924.

[52] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. Ddos in the iot: mirai and other botnets. *Computer*, 50, 7, 80–84.

[53] W. Labidi, M. Sarkiss, and M. Kamoun. 2015. Joint multi-user resource scheduling and computation offloading in small cell networks. In *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 794–801.

[54] Neil Long and Rob Thomas. 2001. Trends in denial of service attack technology. *CERT Coordination Center*, 648–651.

[55] Tom H. Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. 2015. Fog computing: focusing on mobile users at the edge. (2015). arXiv: 1502.01815 [cs.NI].

[56] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang. 2018. Selective offloading in mobile edge computing for the green internet of things. *IEEE Network*, 32, 1, 54–60. ISSN: 1558-156X. DOI: 10.1109/MNET.2018.1700101.

[57] Danny McPherson, R Dobbins, M Hollyman, C Labovitzh, and J Nazario. 2010. Worldwide infrastructure security report, volume v. *Arbor Networks*.

[58] E. Mills. 2012. Doj, fbi, entertainment industry sites attacked after piracy arrests, *2012. Available: http://news.cnet.com/8301-27080*$_3 - 57362279 - 245/doj - fbi - entertainment - industry - sites - attacked - after - piracy - arrests$..

[59] Jelena Mirkovic and Peter Reiher. 2004. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34, 2, (April 2004), 39–53. ISSN: 0146-4833. DOI: 10.1145/997150.997156. https://doi.org/10.1145/997150.997156.

[60] A. Mishra, B. B. Gupta, and R. C. Joshi. 2011. A comparative study of distributed denial of service attacks, intrusion tolerance and mitigation techniques. In *2011 European Intelligence and Security Informatics Conference*, 286–289.

[61] Paul Mockapetris et al. 1987. Domain names-concepts and facilities.

[62] Jose Nazario. 2007. Blackenergy ddos bot analysis. *Arbor*.

[63] Alexander Gutnikov Oleg Kupreev Ekaterina Badovskaya. 2020. Ddos attacks in q1 2020. https://securelist.com/ddos-attacks-in-q1-2020/96837/.

[64] Alexander Gutnikov Oleg Kupreev Ekaterina Badovskaya. 2020. Ddos attacks in q2 2020. https://securelist.com/ddos-attacks-in-q2-2020/98077/.

[65] Vern Paxson. 2001. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31, 3, (July 2001), 38–47. ISSN: 0146-4833. DOI: 10.1145/505659.505664. https://doi.org/10.1145/505659.505664.

[66] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. 2007. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, 39, 1, (April 2007), 3–es. ISSN: 0360-0300. DOI: 10.1145/1216370.1216373. https://doi.org/10.1145/1216370.1216373.

[67] M Reza Rahimi, Jian Ren, Chi Harold Liu, Athanasios V Vasilakos, and Nalini Venkatasubramanian. 2014. Mobile cloud computing: a survey, state of art and future directions. *Mobile Networks and Applications*, 19, 2, 133–143.

[68] F. Y. Rashid. 2012. Ddos attack tools, service help target organizations: arbor networks, *Feb, 2012. Available at: http://www.eweek.com/c/a/Security/DDoS-Attack-Tools-Service-Help-Target-Organizations-Arbor-Networks-763865*.

[69] V KRISHNA Reddy and D Sreenivasulu. 2016. Software-defined networking with ddos attacks in cloud computing. *International Journal of innovative Technologies (IJIT)*, 4, 19, 3779–3783.

[70] R. R. Rejimol Robinson and C. Thomas. 2012. Evaluation of mitigation methods for distributed denial of service attacks. In *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 713–718.

[71] J. Ren, Y. Zhang, K. Zhang, and X. Shen. 2015. Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions. *IEEE Communications Magazine*, 53, 3, 98–105.

[72] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. 2018. Mobile edge computing, fog et al.: a survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 78, 680 –698. ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2016.11.009`. `http://www.sciencedirect.com/science/article/pii/S0167739X16305635`.

[73] Dimas Satria, Daihee Park, and Minho Jo. 2017. Recovery for overloaded mobile edge computing. *Future Generation Computer Systems*, 70, 138 –147. ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2016.06.024`. `http://www.sciencedirect.com/science/article/pii/S0167739X16302096`.

[74] M. Satyanarayanan. 2017. The emergence of edge computing. *Computer*, 50, 1, 30–39.

[75] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. 2009. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8, 4, 14–23.

[76] N. Schweitzer, A. Stulman, A. Shabtai, and R. D. Margalit. 2016. Mitigating denial of service attacks in olsr protocol using fictitious nodes. *IEEE Transactions on Mobile Computing*, 15, 1, 163–172.

[77] Arbor Sert. 2011. Ddos and security reports: the arbor networks security blog, *2011. Available at: http://ddos.arbornetworks.com/2012/02/ddos-tools/*.

[78] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, M. Rajarajan, and R. Buyya. 2017. Combating ddos attacks in the cloud: requirements, trends, and future directions. *IEEE Cloud Computing*, 4, 1, 22–32.

[79] Stephen Specht and Ruby Lee. 2003. Taxonomies of distributed denial of service networks, attacks, tools and countermeasures. *CEL2003-03, Princeton University, Princeton, NJ, USA*.

[80] LD Stein and JN Stewart. 2002. The world wide web security faq, version 3.1. 2, february 4, 2002. (2002).

[81] Luis M. Vaquero and Luis Rodero-Merino. 2014. Finding your way in the fog: towards a comprehensive definition of fog computing. *SIGCOMM Comput. Commun. Rev.*, 44, 5, (October 2014), 27–32. ISSN: 0146-4833. DOI: `10.1145/2677046.2677052`. `https://doi.org/10.1145/2677046.2677052`.

[82] Yating Wang, Ray Chen, and Ding-Chau Wang. 2015. A survey of mobile cloud computing applications: perspectives and challenges. *Wireless Personal Communications*, 80, 4, 1607–1623.

[83] David C Wyld, Michal Wozniak, Nabendu Chaki, Natarajan Meghanathan, and Dhinaharan Nagamalai. 2011. *Trends in Network and Communications: International Conferences, NeCOM 2011, WeST 2011, and WiMON 2011, Chennai, India, July 15-17, 2011, Proceedings*. Volume 197. Springer.

[84] Y. Xie and S. Yu. 2009. Monitoring the application-layer ddos attacks for popular websites. *IEEE/ACM Transactions on Networking*, 17, 1, 15–25.

[85] R. Yu, J. Ding, S. Maharjan, S. Gjessing, Y. Zhang, and D. H. K. Tsang. 2018. Decentralized and optimal resource cooperation in geo-distributed mobile cloud computing. *IEEE Transactions on Emerging Topics in Computing*, 6, 1, 72–84.

[86] R. Yu, X. Huang, J. Kang, J. Ding, S. Maharjan, S. Gjessing, and Y. Zhang. 2015. Cooperative resource management in cloud-enabled vehicular networks. *IEEE Transactions on Industrial Electronics*, 62, 12, 7938–7951.

[87] S. Yu, Y. Tian, S. Guo, and D. O. Wu. 2014. Can we beat ddos attacks in clouds? *IEEE Transactions on Parallel and Distributed Systems*, 25, 9, 2245–2254.

[88] S. T. Zargar, J. Joshi, and D. Tipper. 2013. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE Communications Surveys Tutorials*, 15, 4, 2046–2069.

[89] Marvin Zelkowitz. 2005. *Advances in Computers: New Programming Paradigms*. Elsevier.