MASTER THESIS

Middleboxes Against DDoS Attacks

Conducted within the
Information Science and System Management
Programm at the
University of Applied Sciences, Salzburg

submitted by:
Alexander Prestele, BSc

Studiengangsleiter:	FH-Prof. DI Dr. Gerhard Jöchtl
Betreuer:	FH-Prof. Dr. Ing. (habil) Ulrich Hofmann

Salzburg, October 2013

Declaration of Academic Honesty

Herewith, I declare that I have written the present bachelor thesis fully on my own and that I have not used any other sources apart from those given.

Salzburg, the October 15, 2013

	1110581007
Alexander Prestele, BSc	Matrikelnumber

General Information

First- & Last Name:	Alexander Prestele, BSc
Institution:	University of Applied Sciences, Salzburg
Course of Study:	Master in Information Science and System Management
Title of the Thesis:	Middleboxes Against DDoS Attacks
Keywords:	DDoS, Application Layer Routing, OPNET Simulation, P2P CDN Combination, Networkarchitecture
Supervisor:	FH-Prof. Dr. Ing. (habil) Ulrich Hofmann

Abstract

DDoS attacks become more and more sophisticated over the time as the recent attacks show. Attacks with strength of 20-30GB/s of Traffic are not uncommon anymore. Therefore new ways of countermeasures have to be invented, which offer the counterparty an option to react to a DDoS attack in a fast and easy way. This goal is being achieved in this work by combining the common Content Delivery Network (CDN) with the Peer-to-Peer (P2P) technology. The work contributes to the usage of middleboxes as DDoS counter-measure.
The object of this work is to analyze and develop a concept of utilizing the Technology of Application Layer Routing, in Specific P2P routing, as a countermeasure against DDoS attacks. After an overview about the technical fundamentals, an anti DDoS architecture is being presented. This theoretical architecture is being implemented in the simulation software OPNET. The results of this simulation are shown and discussed in the summary of this work.

Acknowledgement

I would like to thank my mother, my father, my two brothers as well as my girlfriend for their ongoing support.
Also i would like to thank my supervisor FH-Prof. Dr. Ing. (habil) Ulrich Hofmann for his strong support.
For the opportunity to go abroad and the organization i would like to thank Mr. Steve Mullins, MBA, and Prof. Alexander Bordetsky from the Naval Postgraduate School as well as the Marshall Plan Foundation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> *"'Security against defeat implies defensive tactics; ability to defeat the enemy means taking the offensive"'* [1].

This is how Sun Tzu describes the way of defeating an enemy in his book *The Art of War*. These thoughts, which were written in the year 512 BC, are still used by strategists and managers. The fight against the opponent is therefore based on a good defense strategy. But to be able to defeat the enemy you need to take certain offensive measures. Defeating in this context means, that the basis is that strong that it can't be overrun and therefor strengthens the offensive.
Distributed Denial of Service (DDoS) attacks are currently a big threat for firms, government facilities and the military. Due to the synchronous DDoS attacks of the group "'Anonymous"' against MasterCard, VISA and PayPal, their services were not reachable anymore. This action was only possible, because many volunteers formed a botnet by using the program *Low Orbit Ion Cannon*. The tactic of utilizing a botnet as part of a DDoS attack is by now the most used way of performing a DDoS attack. This method enables the opponent to attack even well guarded systems. In 2013 the assaults reached a hitherto unreached level. The Website *Spamhaus* was confronted with a 300GBit/s strong DDoS attack. Even though they had, due to their history, a lot of experience with these kind of attacks, they were not able to defend themselves. Against such sophisticated attacks are even modern security systems unprepared. Therefore new ways have to be found to encounter these strong and massive attacks.
This thesis gives an overview about the different kinds of DDoS attacks and the most common countermeasures. Next, a DDoS attack is simulated within the sim-

ulation environment OPNET. Based on this simulation a counter-measure based on Application Layer Routing is presented.
It is possible to create a network topology by using ALR which has the following features:

- easy scalable
- decreases server load
- maximum content deployment and thereof resultant redundancy
- harder to attack

Chapter 2

Denial of Service

The main goal of a DoS attack is to compromise a service or make it unreachable. The methods that are used to start a DDoS attack are most of the time pretty simple, nevertheless very effective. The current model of the internet with its possibility to reach every server is used against itself. The different types of attacks makes it hard to defend against every kind of attack. Following points are valid for every PC or server, which is connected to the internet [2], [3], [4], [5].

- Every PC, no matter how well protected, can be a victim of a DDoS Attack [6].
- Every host or hardware, which is used in the way of the communication, has a certain limit, which can be maximal exhausted [6].
- The intelligence of the internet is always at the endpoints not in the network itself. The internet is built for fast data transmission and can therefor be misused by everyone, because security systems are not implemented in the middle of the network [6].

2.1 Botnet

A botnet is a combined attack of many PCs to be able to attack even big infrastructures. Such big infrastructures are getting more and more capable of handling large amount of requests at a time. Therefor attackers use so called *Zombies* or *Agents*. To be able to control these Agents, PCs were previously infected by malware. Most

of the time, a so called *Zero-Day-Exploit* is used. These are software weak points, which are exploited by attackers right after being published. Most of the time a software patch is not available. This means that the systems remain vulnerable until a patch is released. These infecteded PCs form a so called botnet, where most of the PCs are used as Agents and some as Master. These Master PCs have the task to spread the commands from the attacker to the Agents and to store their IP addresses (see figure 2.1).

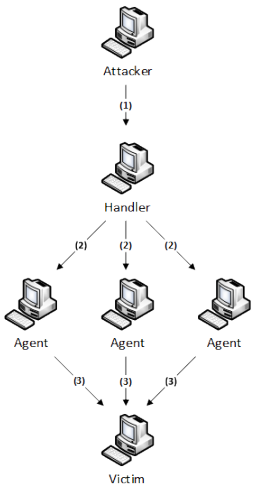

Figure 2.1: Agent-Handler-Model - Structure of a Botnet

The communication between Attacker, Master and Agents is realized by Internet Relay Chat (IRC) channels. These channels are called *Control Traffic*. A basic function of IRC is the possibility to encrypt the control traffic. Even if an Agent is discovered, it just means that perhaps the IRC channel to the master is revealed, which may lead to the discovery of some more Agents, the IRC network itself is still hidden. In this way the exposure of an IRC network is nearly impossible. From now on DoS attacks are always called DDoS attacks, due to their botnet architecture [4], [7], [8].

2.2 DDoS Attacks

The following section describes how DDoS attacks can be classified and what types of attacks are used.

Figure 2.2: Classification of DDoS

2.2.1 DDoS Classification

DDoS attacks can be classified into the following categories, *Architecture*, *Degree of Automatization*, *Impact*, *Exploited Vulnerability*, *Attack Rate Dynamics*, *Propagation Strategy*, *Scanning Strategy* and *Paket Contents*.

Architecture Depending on how the botnet, which is used for the attack, is built, a categorization is possible. The main categories are *Agent-Handler-Model*, *IRC-Based Model* and *Reflector Model*.

Degree of Automatization This point describes the way how and when an attack is started after a PC is infected by malware. The different possibilities of starting an attack are *Manual*, *Semi-Automatic* and *Automatic*

Impact *Disruptive* and *Degrading* are the two Impact results. They describe of what kind the result of an DDoS attack is.

Exploited Vulnerability The exploited vulnerability section is one of the more important ones. The two distinctions of *Bandwitdh depletion* and *Resource depletion*

explain the different tactics to exploit a certain vulnerability.

Attack Rate Dynamics The attack rate of a DDoS attack plays a certain role if and how an attack can be identified. *Continuous* and *Variable* are the two main methods.

Propagation Strategy The propagation strategy describes on how the attack code is transported onto the new clients. The three possibilities are *Central*, *Back-chaining* and *Autonomous*.

Scanning Strategy Finding new PCs and infecting them with malware is a time consuming process, if just one PC is used. Therefor the botnets itself is performing this action to achieve the goal of extension. This task can be fulfilled in four ways. *Random*, *Hitlist*, *Topological* and *Logical Subnet*

Packet Contents Categorize packets by their content divides the packets into *Filterable* and *Non-Filterable*. It is necessary because DDoS attacks are using legal connection requests and illegal ones.

2.2.2 DDoS Types of Attack

The following section gives an overview about the most common DDoS attack.

UDP Flood Attack UDP flood attacks are the easiest way of performing an DDoS attack. The main goal is to overload either the server or the bandwidth towards the victim, to full capacity. In the case of the bandwidth depletion, a flood of UDP packets destined for random ports of an server, will deplete the servers bandwidth. If e.g. the link has a capacity of 10Gb/s and the attacker sends 11Gb/s, the server is not reachable anymore, because the link towards the server is full and packets destined for the the server will be dropped. Depending on the network infrastructure, this attack can even have an effect on other services within the victims network. In the other case of overloading a server, the UDP flood will try to send UDP connection request to random UDP ports of a server. This server then, answers these requests with a "'Destination Port Unreachable"' notification. The goal is that the server is so busy with answering these request, that no time is left for

answering legal requests. Due to the fact that a UDP flood attack can be detected by a firewall, attackers try to hide the attack by faking e.g. the IP addresses of the packets (spoofed IP address) [9], [10], [11].

ICMP Echo Attack The Internet Control Message Protocol (ICMP) is a protocol, which is spoken by every IP capable end device. It is used to send failure notifications or e.g. it is used to find routes in the internet (via traceroute) with the help of the Time-To-Live (TTL) Field. The basis for an attack with ICMP is the ICMP_ECHO_REQUEST (Ping) command. This command requests another servers current status. The originator gets an answer in form of an ICMP_ECHO_REPLY notification. A DDoS attack sends a vast number of ICMP_ECHO_REQUEST to increase the bandwidth utilization as well as the servers workload [9], [10], [12].

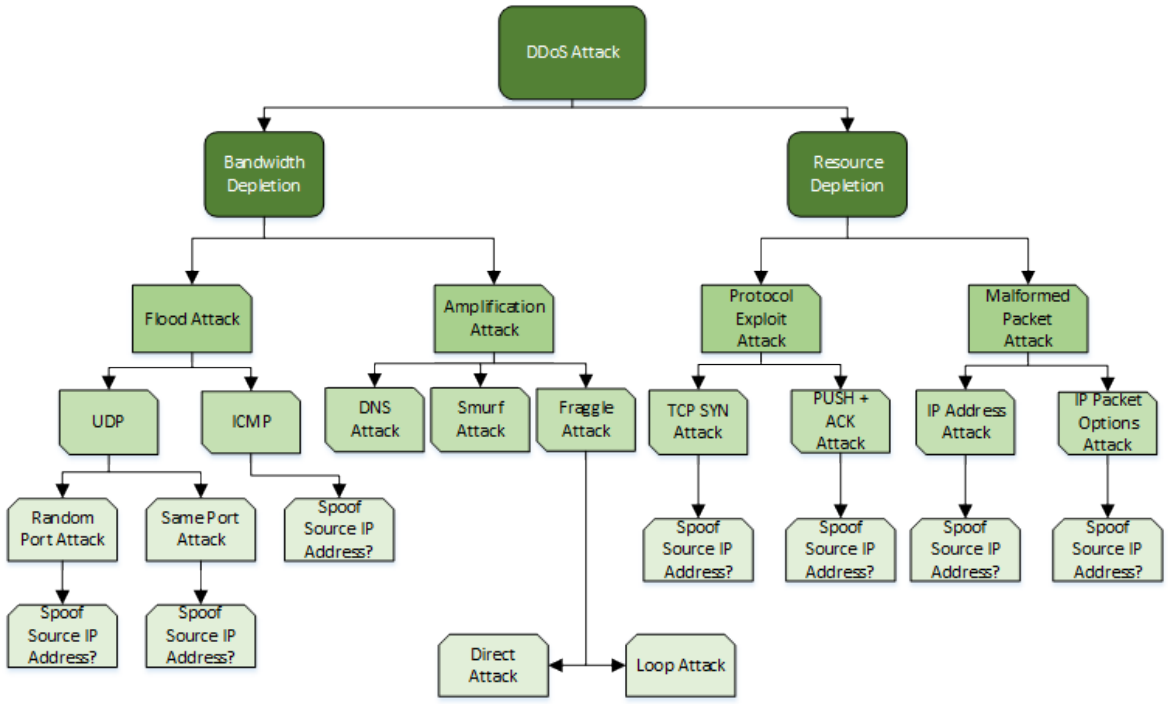

Figure 2.3: Classification of DDoS Attacks

Smurf Attack Smurf is an amplification attack. This type of attack uses the broadcast address of a network to multiply a message. First, the attacker sends a message over the internet to a routers broadcast address. But the message itself has the victims IP address as the source IP address in its header (spoofed IP address). This

router then sends the message to all his clients in his network and all the clients respond the message. But instead of sending the message back to the attacker, they are sending the response to the victim, due to the spoofed header. This way the attacker uses a existing network infrastructure and does not need to infiltrate PCs first. Further, the attacker managed it, that the strength of the attack got multiplied by the number of clients in the routers network. Another point is that the attacker is not attacking the victim itself, but through the other clients. That way he can cover his tracks. The basis for these attack are most of the time ICMP_ECHO_REQUEST which are answered by the clients by sending ICMP_ECHO_REPLY notifications to the victim [9], [12].

Fraggle Attack Fraggle attacks are like Smurf amplification attacks. Only, instead of sending ICMP_ECHO_REQUEST, the attacker is sending UDP ECHO notifications to the victims port 7 (echo-Port) or 19 (chargen-Port) [9].

DNS Reflection Attack This form of an attack is nowadays the most used DDoS attack. Measured on their impact, they are the most severe ones. They belong, like Fraggle and Smurf, to the amplification attack category. In the case of Domain Name System (DNS) reflection, not the number of IP packets is increased, but the size of the IP packets. The basis for this attack are *Open DNS Resolvers*. Open DNS resolver are DNS server who accept request from anyone in the internet. An attacker is requesting e.g. complete Domain Name System Security Extensions (DNSSEC) entries, which are usually very big. The query is possible, because they are established via UDP connections. Therefore no explicit connections has to be established beforehand, neither a security check is performed. That way IP addresses can be spoofed.
The attacker commands a botnet (e.g. via IRC) to perform certain DNS requests. Theses request are destined to such a open DNS resolver. The reply to the request is most of the time much bigger than the request itself. The DNS server sends the replay then to the victim, due to the spoofed IP address. Because of the size and the amount of the IP packets, the capacity of the victims server bandwidth is overburdened. Technically speaking, DNS reflection attack belong to the class of amplification attacks as well as the class of flooding attacks [13], [14], [15].

TCP SYN Flood The TCP SYN flood is part of the *Resource Depletion* attacks. These attacks are exploiting weaknesses in the implementation of protocols by sending packets with intentionally changed headers. The goal is either to exhaust a servers or the networks capacity.
TCP is the most used communication protocol in the internet. The attack exploits the *Three-Way-Handshake*, which TCP uses. In a normal establishing of a TCP connection, a client sends a SYN packet to a server. The server answers the SYN request with an ACK and sends another SYN request to the client, who then confirms the SYN request with a ACK reply. This way, the Three-Way-Handshake is complete an the client can request content from the server. If a Client does not send the last ACK packet, the server will resend the ACK und SYN packet, until the client responds or a timeout occurs. This state is called a half-open connection [16].
TCP SYN attacks exploit the protocol in that way, that they will never send the last ACK notification.The first two steps are identical with the normal Three-Way-Handshake with the difference that a spoofed IP address is used. That way the server sends the ACK and SYN packets to any PC in the internet. The PC, who never requested a TCP connection, discards the packet. The more half-open request the server has, the less resources he has left. At some point the server will not be able anymore to open new connections to legitimate clients and the service is unreachable [9], [10], [16].

PUSH + ACK Attack By default the TCP protocol utilizes buffering of data until a IP packet has the maximal allowed size. This way the overhead, which every header of a new packet creates, is minimized. Time critical applications can ask the server, or in this case the victim, to send the packet imitatively. This is achieved by setting the PUSH field in the TCP header.
PUSH + ACK attacks blong to the class of *Resource Depletion* attacks. The TCP connection between attacker and victim is completely established, other than e.g. in a TCP SYN attack. The difference to a normal TCP connection request is, that the attacker does not only send the last ACK packet, he also sets the PUSH flag. This causes the victim to empty his buffer. Even if the buffer is empty, the victim has to look if there is any data in the buffers queue. This way the victim is occupied with a useless action and can not accept new connection request if a large amount of PUSH + ACK requests are sent [9], [10], [16].

IP Address & IP Packet Options Attack The category of *Malformed Paket Attacks* is part of the *Resource Depletion* attacks. In this category are the so called *IP Address- und IP Options* attacks. The goal is either to exhaust or at least reduce a victims server performance by sending malformed IP packets.
The IP address attack tries to compromise the servers performance by sending packets with the same source and destination header. IP option attacks sets randomly every flag in an IP header. Only die Quality of Service (QoS) flags are all set. This way the victim needs more time to process every packet [9], [10].

2.2.3 DDoS Programs

Most of the programs which are used to perform a DDoS attack are controlled via IRC. On the one hand it is easier to control and hide the botnetwork and on the other hand it protects the attacker from detection. Therefore the following tools are IRC based [9], [10], [17], [18].

Knight The first reports about Knight were in 2001. It supports the most forms of SYN attacks, among them TCP SYN, as well as UDP flood attacks and it has the possibility to create malformed packets. It is typically installed by hiding it in another programs code (Trojan horse). Knight is the basis for other, newer programs like *Kaiten*, which supports newer forms of UDP and TCP flooding and PUSH + ACK attacks. With using Kaiten, the attack can change the hole 32bit of the source IP address (spoofed IP address).

Trinity v3 Trinity v3 is, with its predecessor Trinity, the basis for the most available DDoS programs. On of the Programs which is used for proof of concept demonstration is *Shaft*. Shaft is an improved form of Trinity v3. Trinity v3 supports a bunch of flooding attacks, among them UDP, TCP SYN, TCP ACK and TCP NUL (Portscan). It is possible to change the hole 32bit of the IP source header.

Low Orbit Ion Cannon (LOIC) This software was originally created by Preatox Technologies in C# for network stress tests. After the sourcecode was published, different JAVA versions were created, among them LOIC. LOIC uses TCP and UDP flooding techniques. The group *Anonymous* asked their supporters to install and help them with their DDoS attacks by using LOIC.

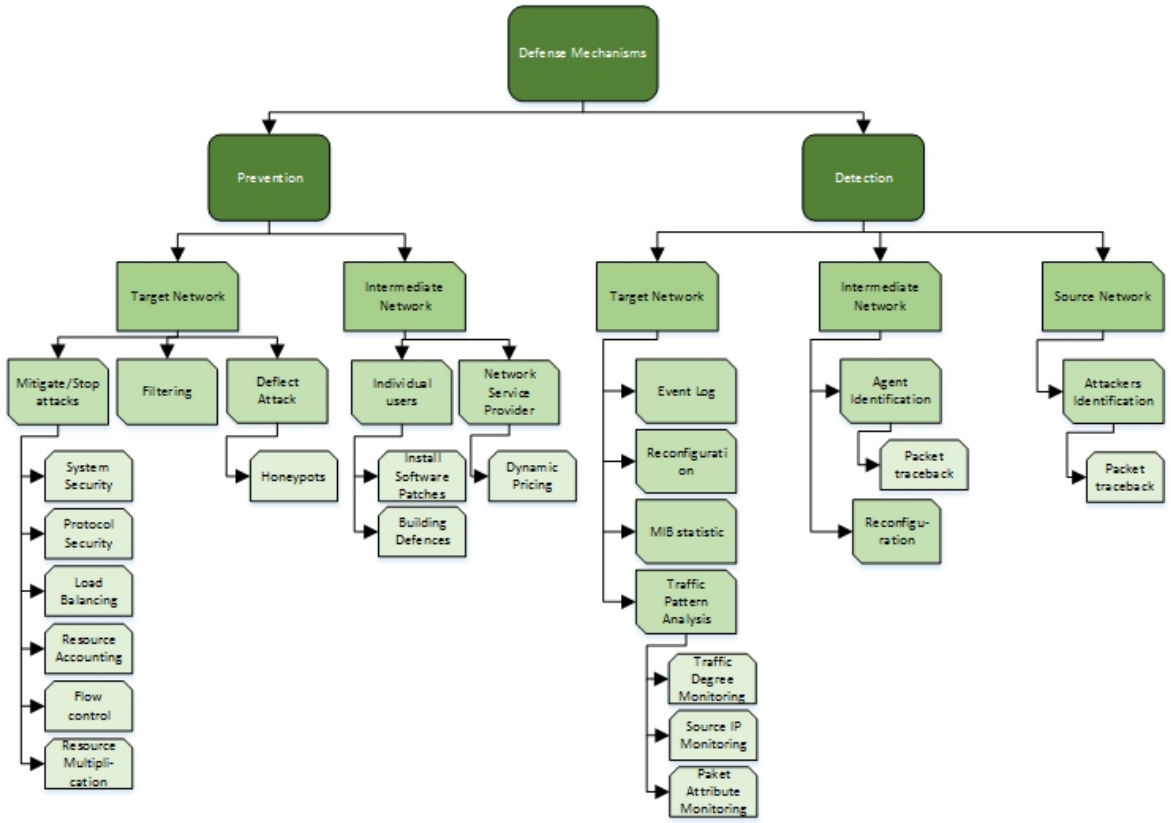

Figure 2.4: Classification of DDoS Counter-Measures

2.2.4 DDoS Counter-Measures

To encounter a DDoS attack both knowledge about the way attacks are performed and the current counter-measures is needed. This is especially important, due to the easy way of performing a DDoS with the use of exploits and metasploits. The following points sums the previous chapter up.

- DDoS attacks generate high flood of data to overwhelm a victim.
- The differentiation between a legal connection request of a normal client and the connection request of an attacker is hard to achieve.
- Most of the DDoS attacks use spoofed IP addresses.
- Due to spoofed IP addresses and the high number of agents which are involved in a DDoS attack, traceback of an attack is nearly impossible.
- Standard firewalls filter only by IP header and not by content. Only *New Generation Firewalls* do a *Deep-Packet Inspection* and can therefore detect fake connection requests. Good spoofed and manipulated packets are still hard to

detect.

- A distributed attack demands a distributed defense. Only the coordination and communication between such defense systems is hard to achieve [19].

Prevention Mechanism The best option to encounter DDoS attacks is prevention. The prevention mechanisms can be in the entry point of the target network or in the network itself. The target network is the actual target. Therefore the most defense systems are installed in this area. This security area is divided in three categories. *Mitigate/Stop attacks*, *Filtering* and *Deflect Attack*.
Mitigate/Stop attacks sums up the points, which are important to the systems security, like *System Security* (improving e.g. physical access to hardware), *Protocol Security* (changing to newer protocols), *Load Balancing* (spreads connection requests e.g. to multiple servers), *Resource Accounting* (tries to achieve fair distribution of e.g. servers performance onto all clients), *Flow Control* (regulates traffic between e.g. a fast packet sending server and slow packet accepting client) and *Resource Multiplication* (basis on strong overdimensioning). The second point, *Filtering*, is the most used one in context of security. Packets, coming from the internet, are filtered by their IP header content. If a packet does not fulfill certain criteria, it is not forwarded into the network, instead it is deleated. This filtering is done by either firewalls or next generation firewalls. The last point, *Deflect Method*, tries to hide the target system by installing so called *Honeypots*. These Honeypots are systems, which are insecure on purpose. An attacker, who targets such a system does not do damage. Instead, the honeypot system is collecting as much information about the attacker as possible.
The second point of the class Prevention is the *Intermediate Network*. One of the most important points is the *User-Awareness*. Every user, client or server which is connected to the internet should have certain security mechanisms. This way a client ensures, that no malware is installed on the system and no unwanted connections are opened. Due to different OS, this task is hard to achieve. Nevertheless, every user should have enough resources to ensure its own safety.

Detection and Defense The purpose of the detection and defense approach is completely different to the prevention mechanisms. In contrast to prevention, detection and defense tries to learn differences between attack signatures and legitimate traffic, to be able to detect attacks. The actual functional principle of Intrusion

Detection Systeme (IDS) and Intrusion Prevention Systeme (IPS) is based on *Data Mining* and *Artificial Intelligence Techniques*. Detection und Defense is divided into three categories. Next to the Target Network security and the Intermediate Network is also the Source Network one part. The target network section tries to detect and adapt to an actual attack. It is based on *log-Files* in which every event of the system is recorded. By analyzing these files it is possible to compare the current attack to existing attack patterns. The Management Information Base (MIB) of e.g. routers, switches, servers, firwalls, etc. are other sources to find intelligence about an attack [10], [20]. By interpreting these information, firewall, IDS and IDS rules are adjusted as well as the behavior of the load balancer and the access control mechanisms. Systems, who monitor the intermediate network, supply the victim with information. This is achieved by mechanisms like *Traceback* or *HopCount* [21], [22]. That way agents can be detected (*Agent Identification*) and disconnected, by the Internet Service Provider (ISP), until their proper functionality is restored (*Reconfiguration*). The source network has, like the intermediate network, the possibility to detect an attacker (*Attacker Identification*) [9], [10], [17], [18], [23].

Chapter 3

Simulation Setup

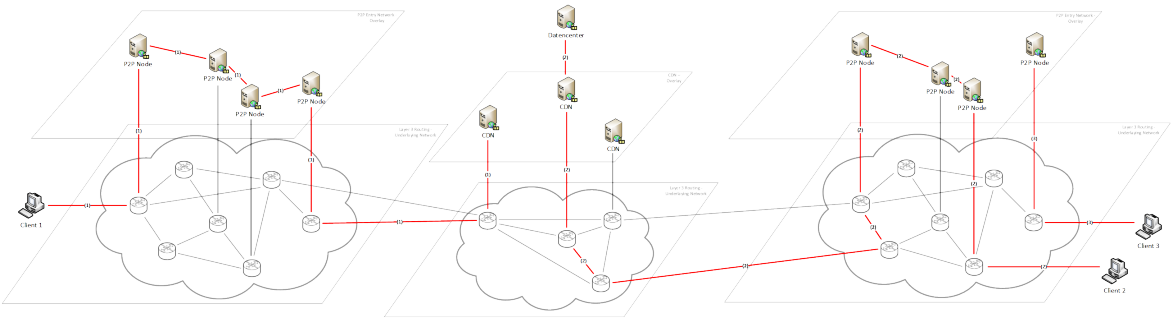

Figure 3.1: Network architecture

To be able to defend an infrastructure against a large amount of DDoS attack types, especially flooding attacks, an architecture based on P2P routing in combination with a classical CDN structure is presented. In addition, this architecture is implemented in OPNET to demonstrate its functionality.
The basis of an attack is most of the time a bandwidth or a servers capacity depletion. P2P is therefore a good counter-measure. On the one hand, bandwidth is still a expense factor, as well as high performance servers, which can withstand a DDoS attack. On the other hand, expenses like *Akamai's* or *Cloudflare's* big datacenters might not be affordable. Not to mention the costs for maintenance, modernization and the constant effort to synchronize the data between different locations. P2P builds a decentralized structure on its own, which also administrates itself. This is done by P2P regarding redundancy and fast access access.
As mentioned before, to be able to defend against large scale DDoS attacks, the data has to be deployed in a decentralized way. By holding and sending data, which has just been requested by a P2P node, every P2P node can send content he has just

yet received. As long as this is done in a way, that every peer gets its content from a local peer, an adequate speed is also guaranteed. As a result, a decentralized content overlay network is created, which can be used above all for static content. Furthermore, dynamic oder security sensitive data have to be accessed through a CDN or normal datacenter, which is a structural content network. For those reasons, a hybrid approach, a combination of P2P and CDN is chosen. This way, by using every servers resources, a large network with low cost is created. Special focus lies on following four points:

- Response time
- Scalability
- Hit rate
- Robustness

A Network has to have certain response times. A network, which can not provide data fast enough is unappealing for the end user. Therefore efficient routing protocols are needed, which can find the searched data from nearby locations. This is only possible, if the data is reasonable replicated and stored. This leads to a higher hit rate. To be furthermore able to defend yourself against DDoS attacks, an easy scalability is also needed. Under a attack it has to adapt fast enough to keep a service alive, even if some nodes fail completely.

3.1 Simulation Architecture

The chapter 2 gave an overview about the most common attack and defense mechanisms. The different layers of the new defense architecture against most of the attacks are presented in the following chapters 3.1.1 bis 3.1.3.

3.1.1 Part 1 - Entry Network

The entry network is the first point were a client enters a network, to find certain data. After the client sent his request to a Local Domain Name System (LDNS), the LDNS returns the IP address of the node which is responsible for the entry network. The DNS has the following functions: He keeps a list of nodes P2P nodes from the P2P network. This list consists of all P2P nodes and their current

workload. Through this list he can determine the node, which is best suited for replying to the request. But not only the current workload is considered, even the locality is a factor. A local node is a geographically short connection between client and P2P node. The advantage is that all the requests afterwards are replied by this node, which results in the best possible latency.
The responsible DNS returns the IP address of the best suited node from the P2P network to the client. The client then requests the data from this node. This node, which is a part of the P2P network, tries to find the data in the P2P network and sends it back to the client (see figure 3.1 - Illustration of different connection types (3)). If the node can not find the information in the P2P network, the node can forward the request to a CDN network, which is only known by the P2P nodes. The node has a sort of a proxy function in this case, due to the fact that he does not return the contact information to the client, but starts a new request. The reply from the CDN is then forwarded to the client by the P2P node as if the P2P node had found it in its network. Furthermore, the node sends the information to the P2P node which is best suited for storing the information. This way the next client request of the same data can be answered faster. The advantage of the P2P nodes behavior are the following. The IP address of the CDN network is only known to the P2P nodes. No client can request data directly from the CDN. Consequently no attack can be started directly on the CDN network, because the CDN network is hidden. Another point is that the CDN knows the P2P nodes already. Even if a attacker finds the CDN network, filtering by source IP addresses is easier. Every packet is dropped, which does not have an IP address from on of the P2P nodes.
A DDoS attack can therefore only concentrate on trying to flood the victim. But in this case, only a P2P node would drop out of the P2P network. The mechanisms, which e.g. *Kademlia* (a P2P protocol) offer, would compensate a breakdown fast and reliable. The P2P network would reorganize itself and the service is still alive and accessible. Even if many local P2P nodes fail, the request could be handled by nodes which are further away. The only result would be a higher latency due to the longer distances. Only an attack, based on normal HTTP requests (like the TCP SYN attack - see chapter 2.2.2) would still work. Depending on the size of the P2P network, the attack can be spread across the hole P2P network which would weaken the attack so much, that the service is still reachable.

3.1.2 Part 2 - CDN Network

The CDN is in between the P2P entry network and the actual datacenter. Its task is to store as much data as possible in its cache, so that the datacenter is only asked rarely for data. This is why nearly all statical content should be stored by the CDN. Furthermore the CDN is the last line of defense for the datacenter. Traffic filtering is a must functionality for the CDN (see figure 3.1 - Illustration of different connection types (1)).
To be able to support the entry network with updates and data, the CDN has to know all the P2P nodes, to categorize them as trustworthy.

3.1.3 Part 3 - Datacenter

The datacenter is the last point were all the data for a client has to be stored. Due to the fact that it is the central point in the architecture, it needs special security measures. This is achieved by the P2P entry network and the following CDN. The datacenter is most of all responsible for dynamical data. Through an efficient reply of request by the P2P network and CDN, the bandwidth which the datacenter needs is low (see figure 3.1 - Illustration of different connection types (2)).

Chapter 4

OPNET

OPNET offers many possibilities to create network simulations as real as possible. Only P2P behavior can not be simulated as how it would work in real. The reason is, that OPNET has no support for dynamical simulations. Due to the fact that this is a central aspect of the architecture, the critical parameter of the P2P network are implemented as if it would be present (see table 4.1). A P2P network realization would be to static that significant information would be simulated. But since the hole simulation is observed for potential realization, a exact implementation is not needed. A exact simulation of the P2P network could be created with a simulation tool like PeerSim [24] or Oversim [25].

4.1 OPNET Implementation

Figure 4.1 shows the global OPNET layer implementation. The global view shows the different locations, in which the clients, P2P nodes and the datacenter are located. The locations are:

New York: Next to monterey the biggest data node with three P2P nodes. The datacenter is also installed here. Average HTTP client number: 1000

Monterey: Three P2P Nodes. Average HTTP client number: 1000

Frankfurt: One P2P node. Average HTTP client number: 6000

Buenos Aires: One P2P node. Average HTTP client number: 1000

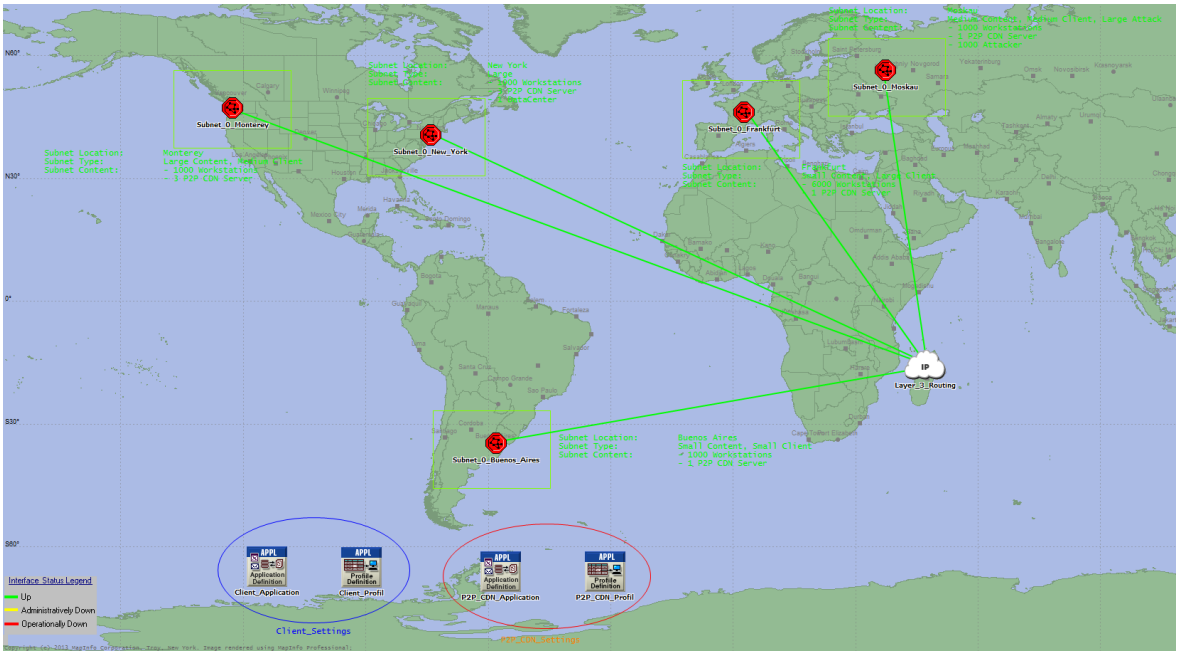

Figure 4.1: Simulation architecture in OPNET: Subnet 0 - Global

Moskau: One P2P node. Average HTTP client number: 1000

Every location is connected to each other by a layer 3 routing cloud. This cloud represents the internet and simulates packet-switched routing, like it is done in the internet.
The underlying level, the subnet 1 view of e.g. Monterey, can be seen in figure 4.1. Within every subnet, every client, the P2P nodes and the datacenter are connected by fiber cables via a router with the gateway. The gateway on the other side is connected to the internet. The CDN nodes are not explicitly implemented as a server, because they just represent another HTTP request. For every 500th HTTP request, representing a HTTP client request which can not be answered by the P2P network itself and is therefore forwarded into the CDN, a delay of 0,05 seconds is used. The databank traffic which arises to update the three CDN nodes can be neglected. To test the functionality of the databank updates the datacenter itself is implemented. This is the point from which the P2P nodes are updated (see figure 4.3.
The selection, which P2P node is responsible for which client is determined by load balancing. The client tries primarily to connect itself to the closest/local P2P node. If another P2P node has a lower workload as the closest one, this P2P is chosen instead. This way the shortest processing time is guaranteed.

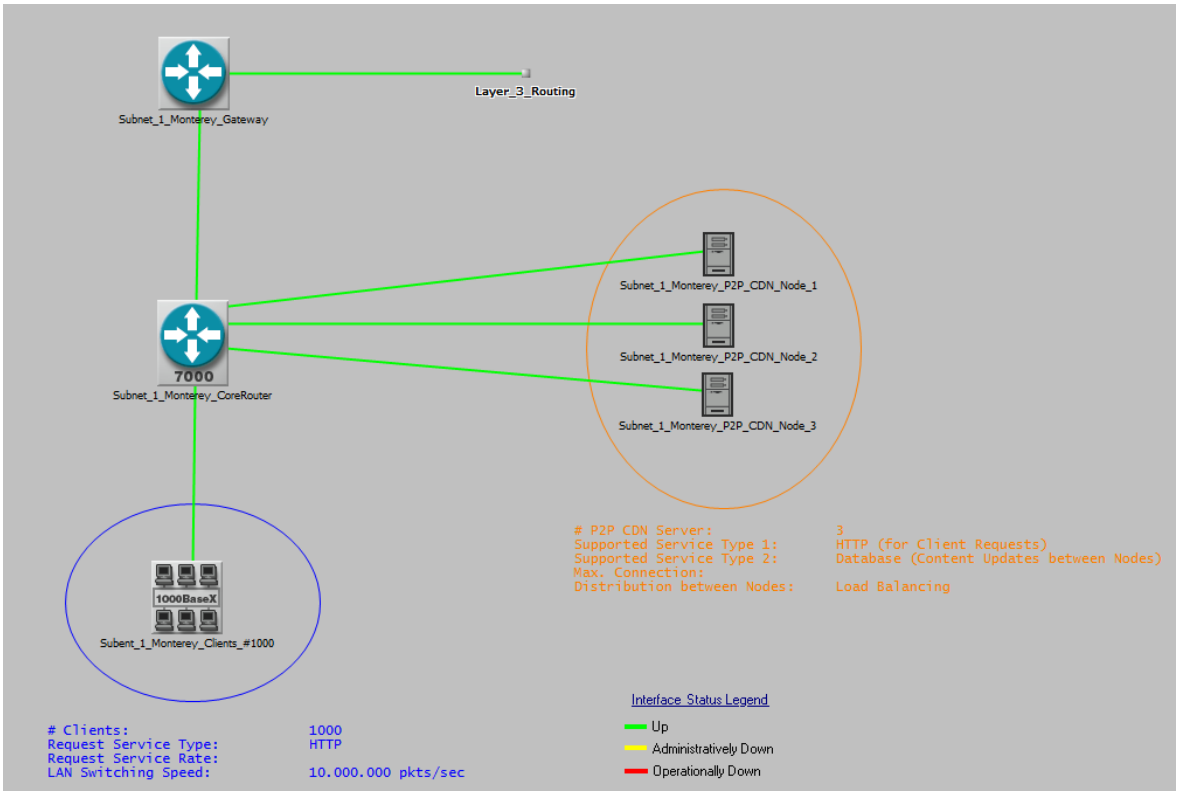

Figure 4.2: Simulation architecture in OPNET: Subnet 1 - Monterey

4.2 Simulation Parameter - Standard Simulation

The stadard simulation has the following parameters (see table 4.2). Every P2P

Parameter	Werte
Point of reference, latency complete site [s]	<1
Point of reference, first object [s]	<0,2
Number of clients (C)	10000
Rate of client requests [per s] ($C_{Request}$)	10000
Number of Services (S)	100
Number of Objects [per Service] (S_{Object})	100
Number of P2P server (P)	9
Rate of P2P Databank updates requests between each other [per s] ($P_{Request}$)	5
Rate of P2P HTTP requests to CDN [per s] ($P\_HTTP_{Request}$)	20
Delay for P2P HTTP requests to CDN Nodes [s]	0.05
Simulation time [h] (T_{Simul})	2

Table 4.1: Simulation parameter - Standard simulation without DDoS attack

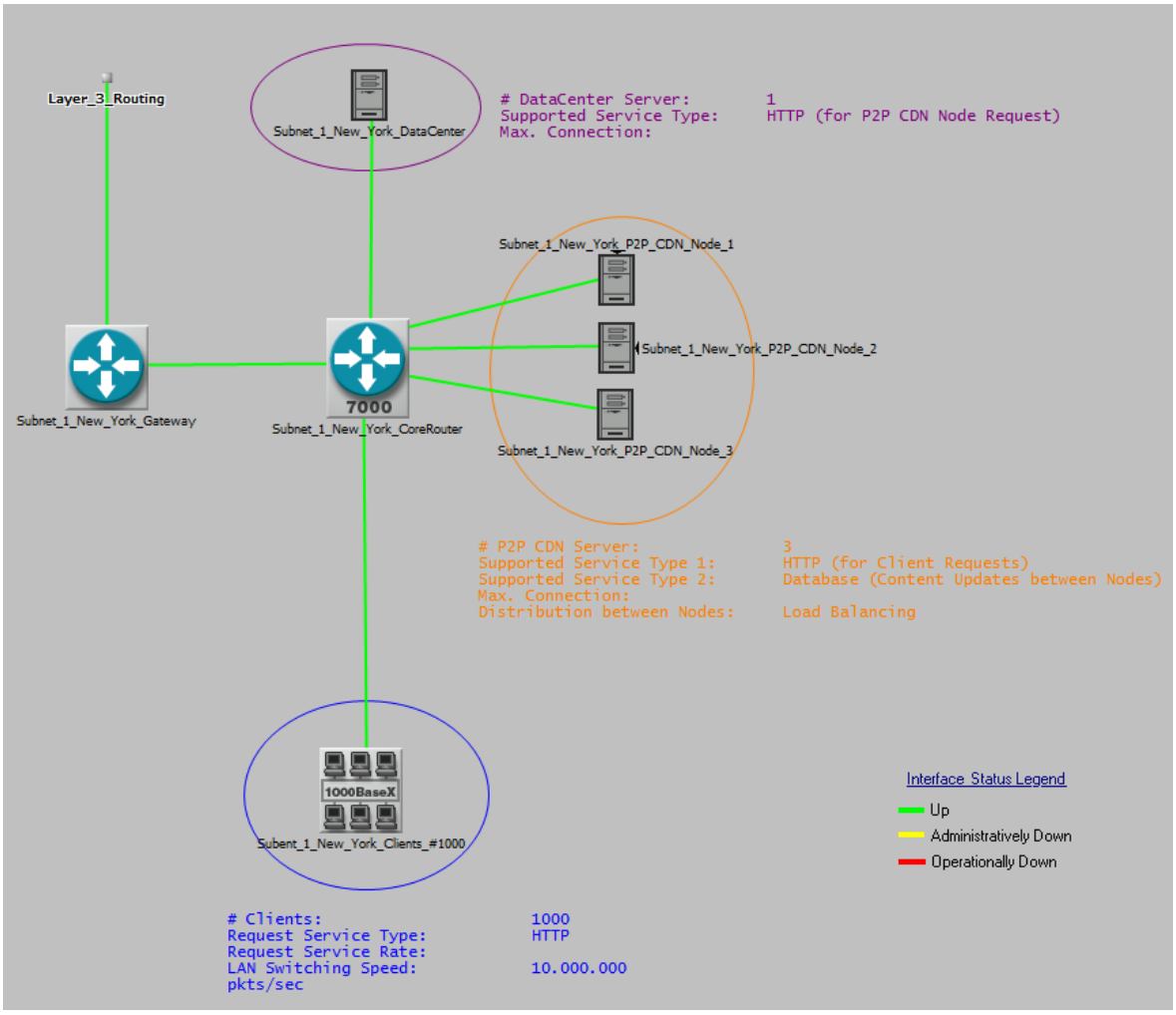

Figure 4.3: Simulation architecture in OPNET: Subnet 1 - New York

node (P), who gets a request from a client (c), should be able to answer this request within less than a second. This point of reference for the simulation is used to give a statement about if the site can be accessed without adverse effects and if the user has a pleasant surf comfort. Within this time, the transmission and the processing of the request has to be done. Ideal response times of the first site object in CDNs are according to [26], [27], [28] under 200ms.

Every node has 100 services, e.g. a website or a video. Every one of these services has 100 objects, e.g. flash plug-ins, pictures, texts and so on. Therefore every P2P Node has $100 \times 100 = 10000$ ($S \times S_{Object}$) objects, which a client can access. The maximum number of P2P nodes is set to 9. The P2P nodes update each other regularly in certain intervals by sending P2P update notes ($P_{Request}$) to store e.g. data on the right node. This depends on the P2P protocol which is used. Also the P2P nodes update themselves in certain intervalls ($CDN\_DB_{Request}$) to have the

newest version of the stored data.
Furthermore it is assumed that every server (or to be precise, the P2P nodes and the datacenter) have enough storage and performance, to store the different contents as well as having enough performance left to process every incoming HTTP request without big delays. Also it is assumed that all the connections between e.g. the nodes, the servers, the clients are dimensioned in a way, that there will be no bandwidth shortage. Therefore a bottleneck effect is not to be expected at any time.

4.3 Simulation Parameter - DDoS Simulation

The DDoS simulation uses the former simulation as basis (see chapter 4.2). Also the parameter for the simulation are the same (see table 4.1) and will only be extended by the DDoS parameters which will be needed for the simulation of the attack (see table 4.2). The high attack rate of 20-30 GB/s has been been selected to cover

Parameter	Werte
DDoS attack rate [GB/s]	20-30
Simulation time [h] (T_{Simu2})	2

Table 4.2: Simulation parameter - DDoS simulation

two factors. On the one side a bandwidth shortage is achieved which corresponds to a bottleneck effect (Flooding attacks - see chapter 2.2.2). The connection to the server, who gets attacked, is fully occupied. This results in a overflow of the routers buffer which is placed before the attacked server. This router starts therefore with discarding packets from the queue. Due to this, legitimate HTTP request can not be differentiated from illegal ones, because they can't even reach the instance which would do the filtering. Even if the request would reach the server, the server could not respond to it because of its high workload (e.g. TCP SYN attack - see chapter 2.2.2).
The simulation time is, like in the standard simulation, limited to two hours. This timespan is enough to give the system on the one hand enough time to adjust and on the other hand it is enough to observe potential extreme states.
The implementation of the DDoS attack is simulated in the OPNET subnet 1 layer of Moskau (see figure 4.4). Every one of the 1000 attacking client executes ping as well as HTTP requests. However those request are performed in a interval of 0.008

seconds. The packet size of a ping request is the maximum allowed size of 65535 Bytes.

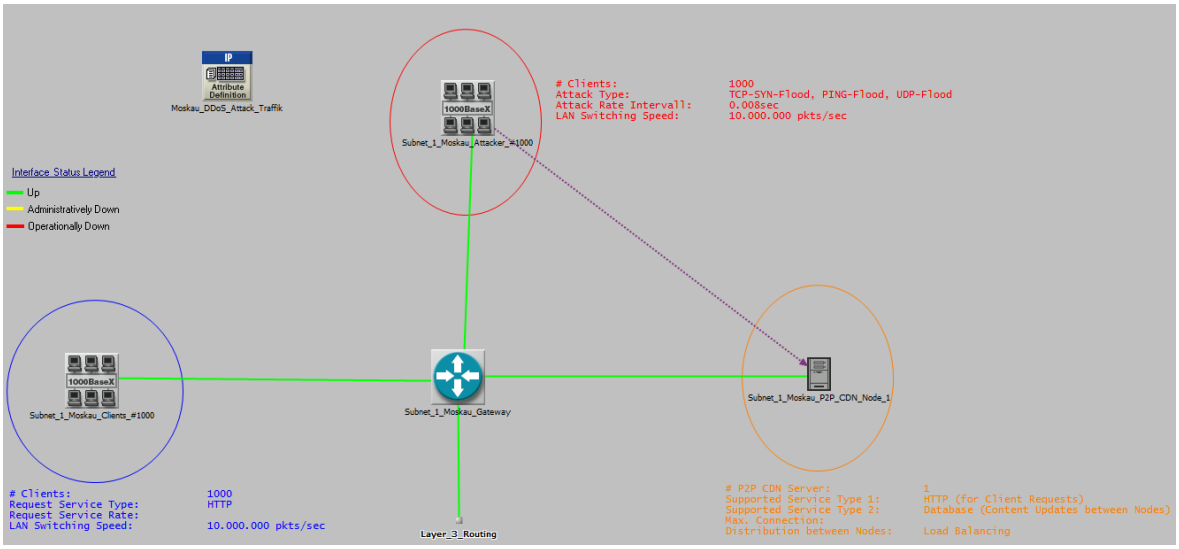

Figure 4.4: DDoS implementation in OPNET: Subnet 1 - Moskau

4.4 Simulation Parameter - Extended DDoS Simulation

The extended DDoS simulation has the following parameters (table 4.3): In this

Parameter	Werte
DDoS attack rate [GB/s]	80-90
Simulation time [h] (T_{Simu2})	2

Table 4.3: Simulation parameter - extended DDoS simulation

simulation, not only the location Moskau is attacked, but also the location Frankfurt and New York. The strength of the attack is tripled and spread over the three location. This results in a non-availability of three of nine P2P nodes.

4.5 Results and Discussion

The following tables show the effects of the simulated DDoS attacks. As a reference value, every figure contains also the simulation results of the standard simulation. The results of the smaller DDoS attack onto the location Moskau is displayed in blue, the ones of the extended DDoS simulation (Moskau, Frankfurt, New York) are in red and that of the standard simulation is in green.
The bandwidth of the e.g. location Moskau are due to the DDoS attack utilized by 100% (see figure 4.5). The utilization of the two DDoS simulations are in blue and that of the standard simulation is in red. Therefore the bandwidth to the P2P

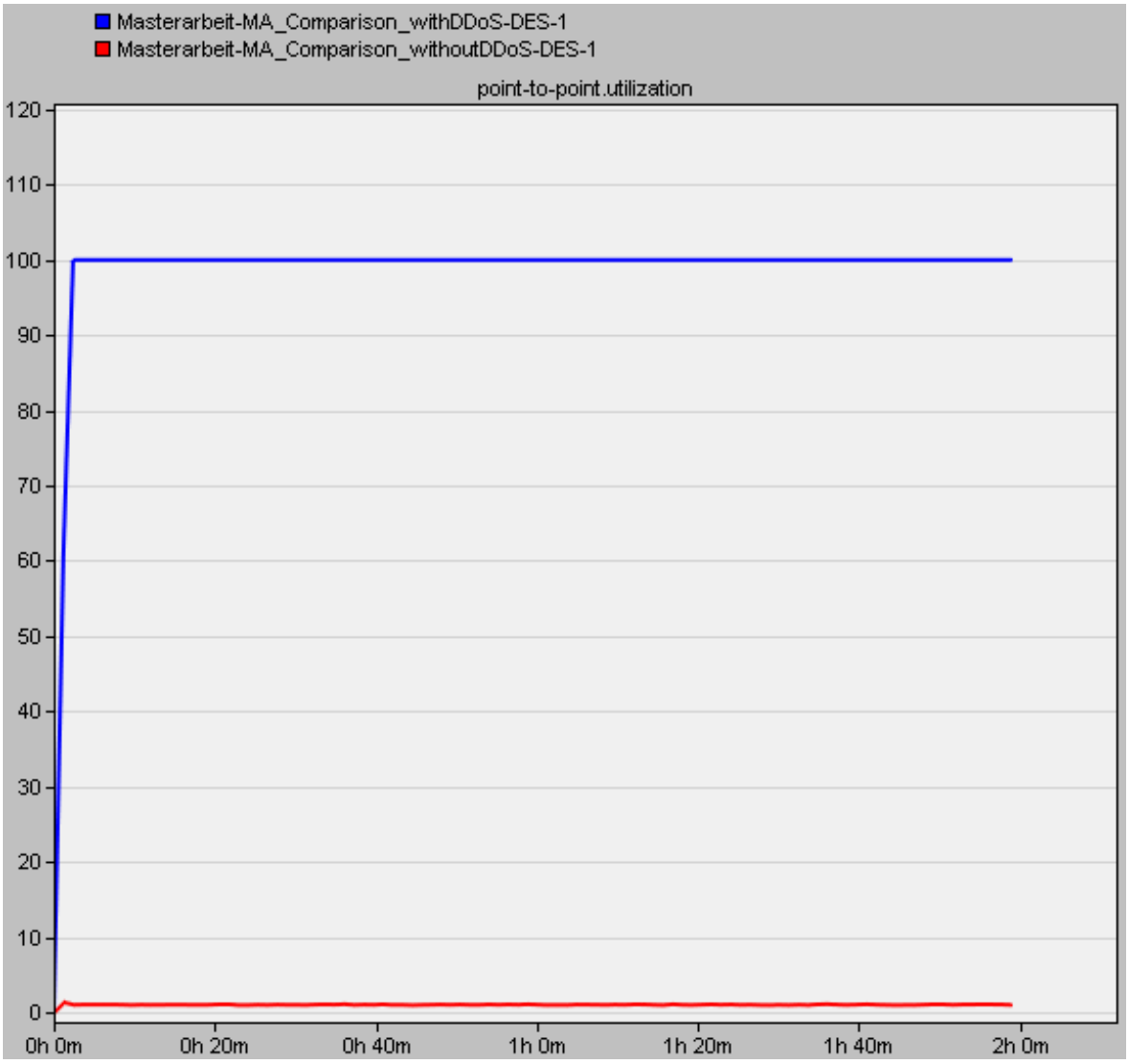

Figure 4.5: Moskau: Bandwidth utilization between CoreRouter and P2P Node

Node is overloaded and no legitimize requests can be received or processed. The

result is a higher timespan which is needed to respond to a clients request (figure 4.6 shows the Moskau clients timespan). This result is exemplary for all clients in the simulation. Despite the DDoS attack, the timespan which is needed to load the

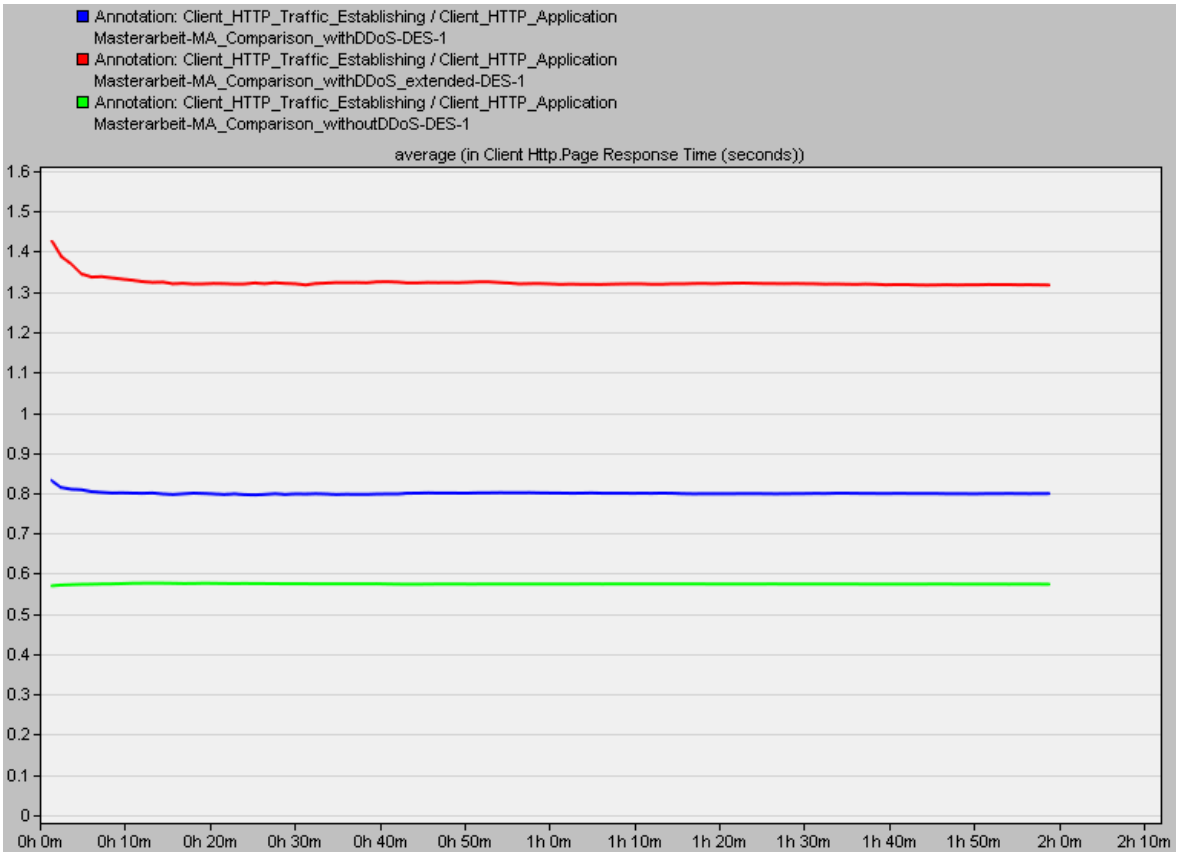

Figure 4.6: Moskau: Client Page Response Time

first object of a Website is within the earlier defined range of under 0.2 seconds (see figure 4.4). The clients from the New York location are the fastest due to the fact that they not only have three P2P nodes in their close surrounding but also the shortest connection to the datacenter. Therefore is is possible to eventually load dynamical content fast. New York is followed by Monterey, which has also three P2P nodes.

These response times are slightly worse than in the standard simulation, in average around 0,1 seconds (4.4). The results from the extended DDoS simulation deviate strongly from the standard simulation. If three from nine P2P node fail, the loading time of a website increases (see figure 4.6). The following figure 4.7 shows the increased loading time of websites, which is characteristically for every location. In this example, the location of Buenos Aires is shown. Through-out the hole simulation an increase in the workload is observable at any node, due to the fact

Nodes	Average Object Response (sec)
Subnet_0_Buenos_Aires	0.177
Subnet_0_Frankfurt	0.165
Subnet_0_Monterey	0.145
Subnet_0_Moskau	0.164
Subnet_0_New_York	0.143

Table 4.4: Average object response time while a DDoS attack simulation

Nodes	Average Object Response (sec)
Subnet_0_Buenos_Aires	0.277
Subnet_0_Frankfurt	0.251
Subnet_0_Monterey	0.232
Subnet_0_Moskau	0.232
Subnet_0_New_York	0.226

Table 4.5: Standard simulation average object response time

Nodes	Average Object Response (sec)
Subnet_0_Buenos_Aires	0.468
Subnet_0_Frankfurt	0.421
Subnet_0_Monterey	0.399
Subnet_0_Moskau	0.398
Subnet_0_New_York	0.378

Table 4.6: Average object response time while a extended DDoS attack simulation

that the P2P node in Moskau is not reachable anymore and all the HTTP requests have to be spread across the other P2P nodes.
Nevertheless, no big difference between the loading time of a website is noticeable. A small increase of ca. 0,25 seconds in average can be seen while the DDoS attack (see figure 4.8). The standard loading times can bee seen in figure 4.9. Both average times are within the allowed maximum time.

The genral increased times, while the DDoS attack occur can be explained by the fact, that the other accessible P2P nodes have to reply to all the clients request. Besides, due to load balancing measures, a request can be sent to a P2P node which is further away. This way the higher transmission delay for the routing of the request/reply has to be taken into account. The Monterey nodes have e.g. due to the increased requests a higher delay in processing their sessions (In figure 4.11 the two DDoS simulations are in blue and the standard simulation is in red)
The simulation has proven, that the architecture has the potential to withstand an

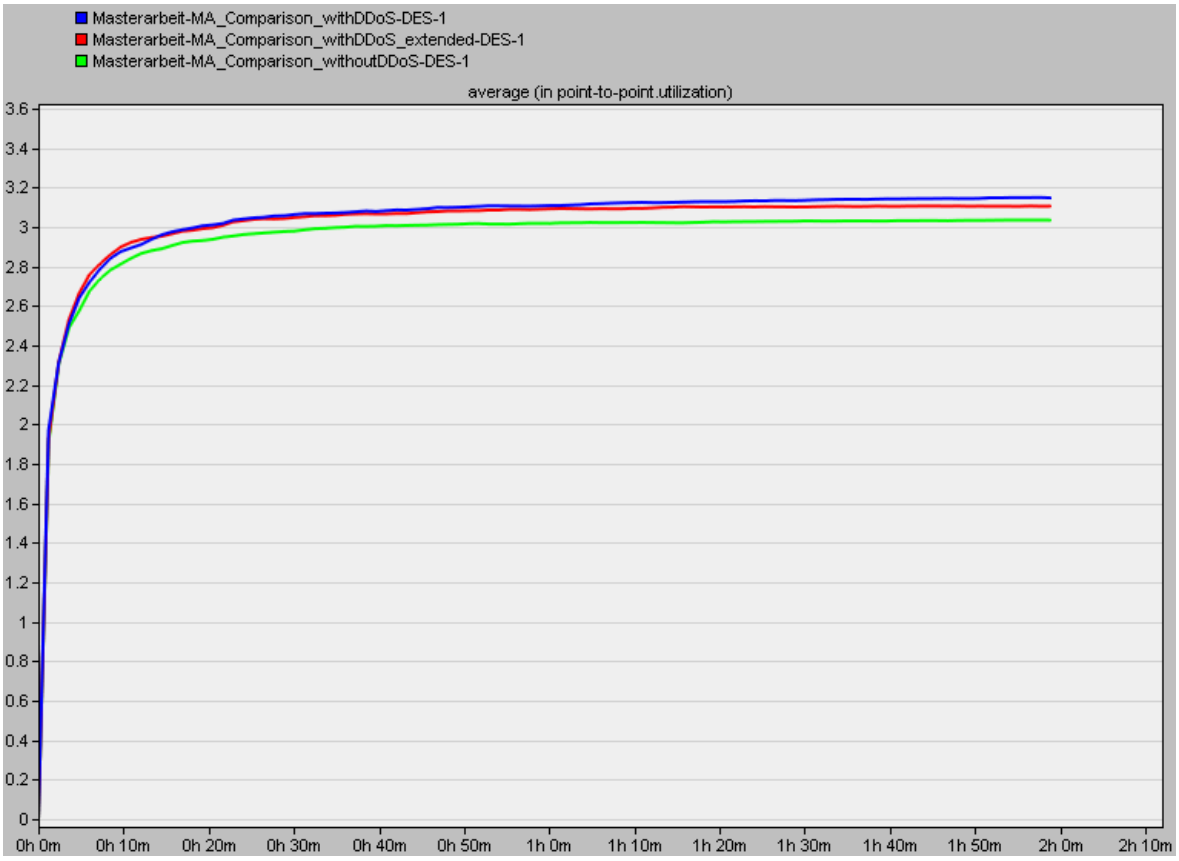

Figure 4.7: Utilization between Layer 3 an the location Standort Buenos Aires in in percentage terms

attack. From the nine P2P nodes in the standard simulation one is not reachable in the first DDoS attack simulation. This does not lead to a huge interference in the service. The other P2P nodes were always able to reply to the requests of the 10000 clients.

In the extended attack, in which three of the nine P2P nodes where unreachable, the loading times got noticeable worse. However to start such an attack a significant higher effort has to be made. To double the loading time of a website it was necessary to triple the attack rate in the simulation.

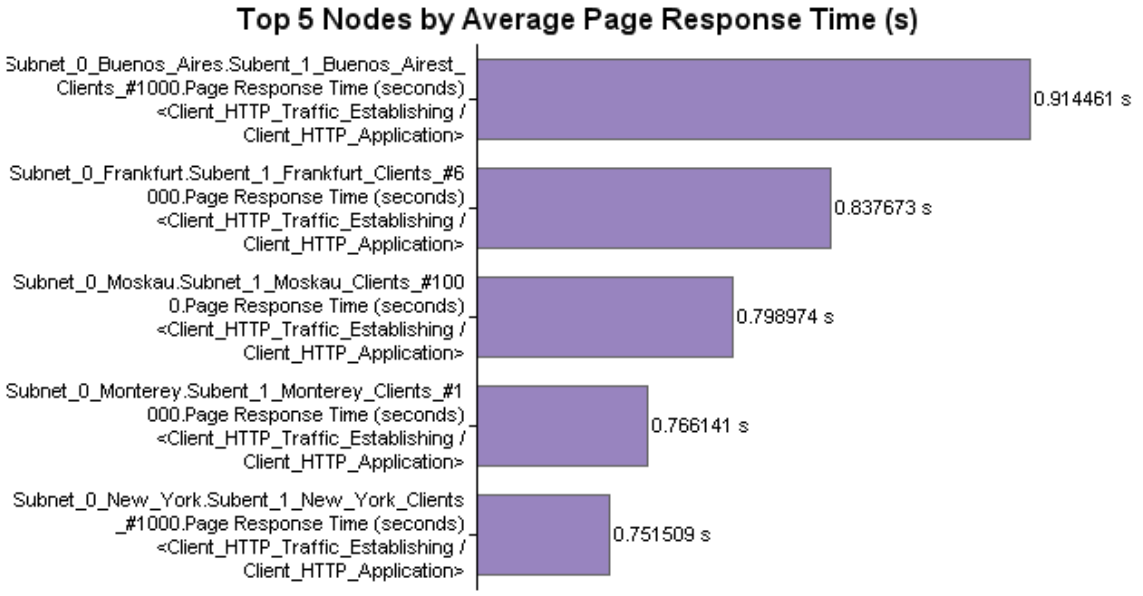

Figure 4.8: Page Response Time while a DDoS attack

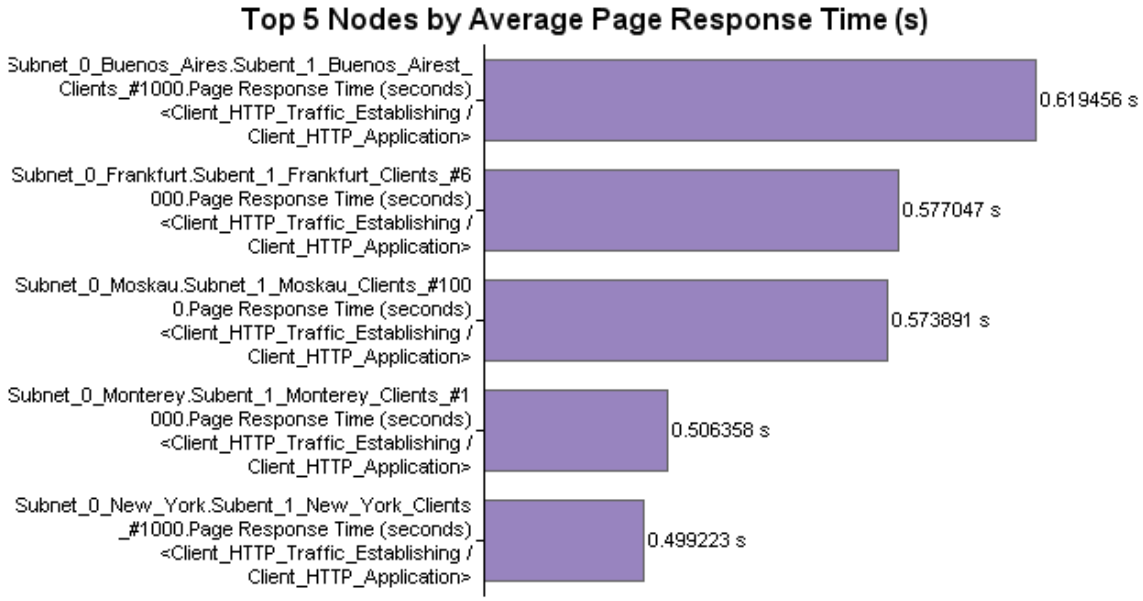

Figure 4.9: Standart simulation page Response Time

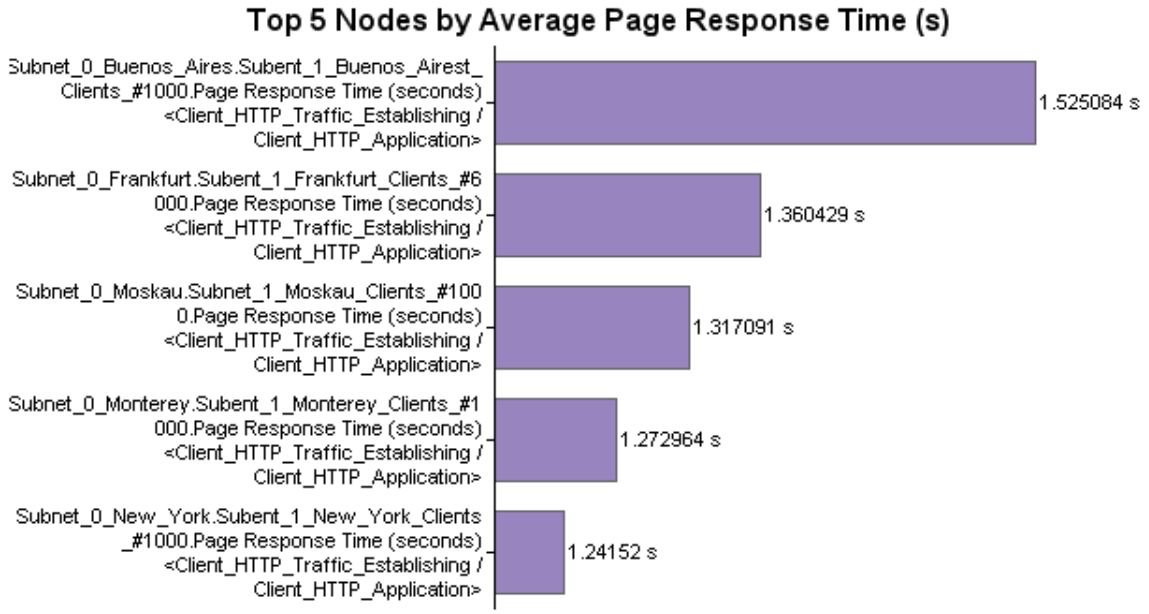

Figure 4.10: Page Response Time while the extended DDoS simulation

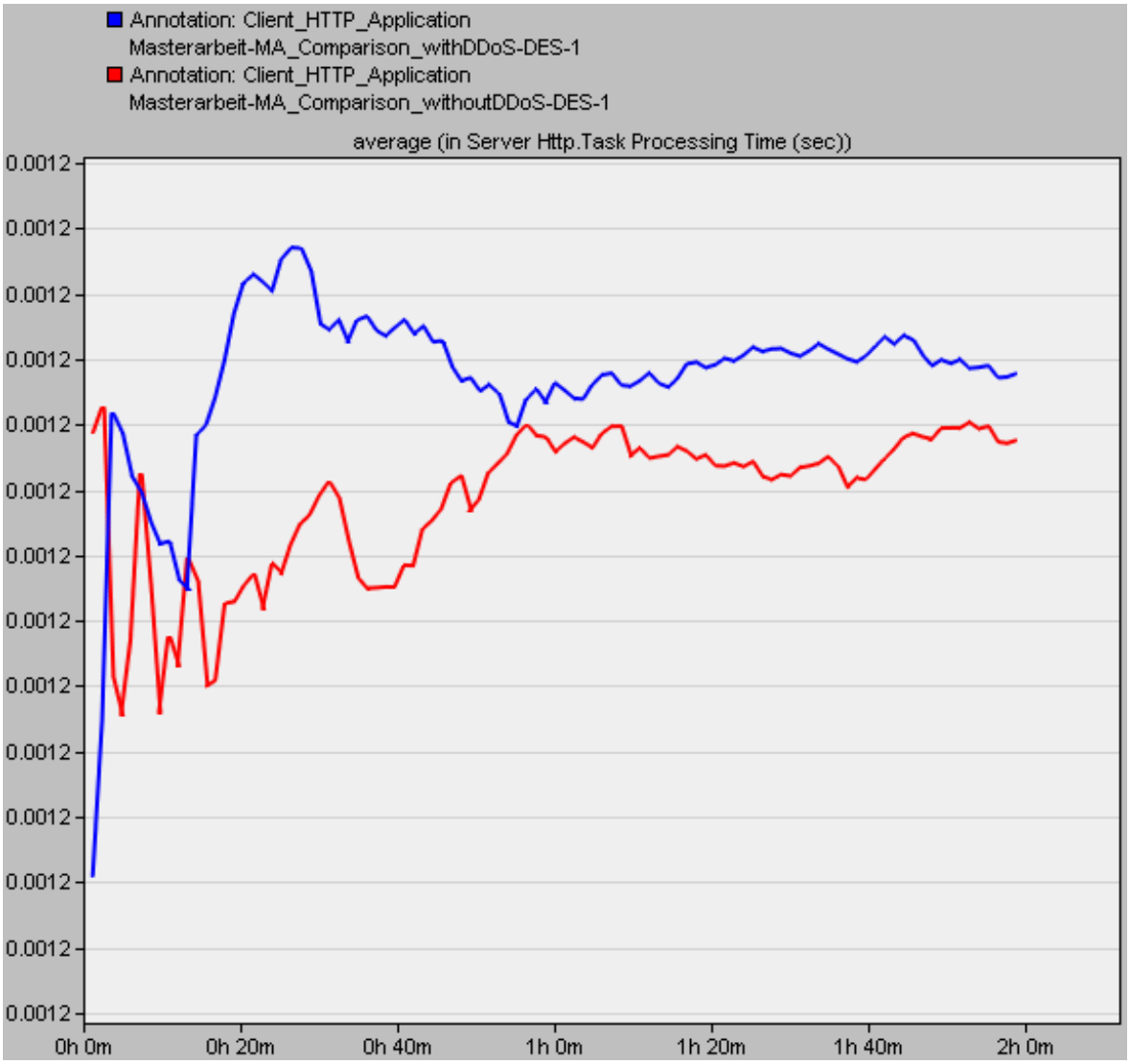

Figure 4.11: Monterey: HTTP processing delay

Chapter 5

Summary

In theory the architecture has proven that it can handle high loads as well as withstanding DDoS attacks. Following table 5.1 summarizes the collected information:

Advantages	Disadvantages
Datacenter is hidden through P2P *"Proxy"* function	Extension needed to detect *"faked"* requests (TCP SYN)
Autonomous replication of data through P2P	
No manual actions needed if node fails	
No manual actions needed if node joins	
Faster filtering as CDN, because P2P nodes are known	
Flooding attacks are not successful	

Table 5.1: Advantages and disadvantages of the P2P CDN architecturee

This kind of architecture could be successful as a open source project. If e.g. multiple owners of servers join each other in a P2P network, the bandwidth could be shared among themselves and DDoS attacks would not be successful. Most of the time the expenses for servers have to be paid as well as the costs for firewalls and other DDoS prevention hardware. Therefor the integration of a P2P entry network could be a advantage, as long as there is enough server and bandwidth capacity

left.
The presented simulation is the ideal basis for future research on the impacts of DDoS attacks or to find counter-measures to such DDoS attacks. E.g. an implementation of CPP would make sense to be able to encounter also TCP SYN attacks. The acquisition of RedSwoosh (P2P content delivery company) in 2007 by the largest CDN operator in the world *Akamai* shows, that P2P has a large potential. Software products like Skype prove, that clients are willing to share their own bandwidth. Therefore a simulation is possible in which the end user has the functionality of the entry network.
Future research could concentrate on the following points:

- Implementation of different filtering techniques at the P2P nodes to find faked HTTP requests.
- Implementation of the P2P architecture in P2P simulation tool (e.g. PeerSim oder Oversim).
- Implementation of clients performing as P2P nodes and compare the results with the, in this work, presented architecture.

Bibliography

[1] S. Wu. *The Art of War*. Dover Pubn Inc, 2002.

[2] C. Douligeris and A. Mitrokotsa. DDoS attacks and defense mechanisms: classification and stateofthe-art. Technical report, Dept. of Inf., Univ. of Piraeus, Piraeus, Greek, Oktober 2003.

[3] J.A. Hamilton, W. Chatam, and J. Rice. Using Simulation to Analyse Distributed Denial of Service Attacks. Technical report, Auburn Univ., März 2010.

[4] S. Ning and Q. Han. Design and implementation of DDoS attack and defense testbed. Technical report, Zhengzhou Inf. Sci. & Technol. Inst., Zhengzhou, China, Dezember 2012.

[5] B. M. Leiner, V. G. Cerf, D D. Clark, R.E. Kahn, L. Kleinrock, D.C. Lynch, J. Postel, L. G. Roberts, and S. Wolff. Brief History of the Internet. Technical report, Internet Society, 2012.

[6] K.J. Houle and G.M. Weaver. Trends in Denial of Service Attack Technology. Technical report, CERT Coordination Center, Oktober 2001.

[7] B. McCarty. Botnets: Big and Bigger. Technical report, Azusa Pacific University, August 2003.

[8] H. Choi, H. Lee, H. Lee, and H. Kim. Botnet Detection by Monitoring Group Activities in DNS Traffic. Technical report, Korea Univ., Januar 2007.

[9] S. Specht and R. Lee. Taxonomies of distributed denial of service networks, attacks, tools and countermeasures. Technical report, Princeton Univ., 2003.

[10] A. Hussain, J. Heidemann, and C. Papadopoulos. A Framework for Classifying Denial of Service Attacks. Technical report, USC/Inf. Sci. Inst., April 2003.

[11] Internet Engineering Task Force. *RFC768 - User Datagram Protocol*, August 1980.

[12] Internet Engineering Task Force. *RFC793 - INTERNET CONTROL MESSAGE PROTOCOL*, September 1981.

[13] CloudFlare. The DDoS That Knocked Spamhaus Offline (And How We Mitigated It). `http://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho` (27.03.2013), 2013.

[14] CloudFlare. How to Launch a 65Gbps DDoS, and How to Stop One. `http://blog.cloudflare.com/65gbps-ddos-no-problem` (25.03.2013), 2012.

[15] M. Prince. Deep Inside a DNS Amplification DDoS Attack. `http://blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack` (14.03.2013), 2012.

[16] Internet Engineering Task Force. *RFC792 - TRANSMISSION CONTROL PROTOCOL*, September 1981.

[17] A. Dr. Asosheh and N. Ramezani. A Comprehensive Taxonomy of DDoS Attacks and Defense Mechanism Applying in a Smart Classification. Technical report, Inf. Tech. Dept., Tarbiat Modares University, Tehran-Iran, April 2008.

[18] J. Mirkovic, J. Martin, and P. Reiher. A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms. Technical report, Comp. Sci. Dept., Univ. of California, Los Angeles, April 2004.

[19] M. Kim and K. Chae. Detection and Identification Mechanism against Spoofed Traffic Using Distributed Agents. Technical report, Dept. of Comp. Sci. and Eng., Ewha Womans University, Korea, April 2004.

[20] J. Mirkovic. *Internet denial of service: attack and defense mechanisms*. Prentice Hall, 2005.

[21] H. Wang, C. Jin, and K.G. Shin. Defense Against Spoofed IP Traffic Using Hop-Count Filtering. Technical report, Dept. of Electr. Eng. & Comput. Sci., Michigan Univ., Ann Arbor, MI, Februar 2007.

[22] M. Sung and J. Xu. IP traceback-based intelligent packet filtering: a novel technique for defending against Internet DDoS attacks. Technical report, Coll. of Comput., Georgia Inst. of Technol., Atlanta, GA, USA, September 2003.

[23] A. Wittmann. DDoS Angriffs und Verteidigungsstrategien. Technical report, Fakultät für Informatik, Technische Univ. München, 2010.

[24] G. P. Jesi. PeerSim: A Peer-to-Peer Simulator. http://peersim.sourceforge.net/ (18.06.2013), 2013.

[25] Karlsruhe Institute of Technology (KIT). Oversim - The Overlay Simulation Framework. http://www.oversim.org/ (18.06.2013), 2013.

[26] P. Maymounkov and D. Mazières. Kedemlia A Peer-to-peer Information System Based on the XOR Metric. Technical report, New York Univ., 2003.

[27] M. J. Freedman and D. Mazières. Sloppy Hashing and Self Organizing Clusters. Technical report, New York Univ., Dept. of Comp. Sci., 2003.

[28] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. Technical report, Microsoft, 2007.