

# Simulation of Internet DDoS Attacks and Defense

Igor Kotenko and Alexander Ulanov

St. Petersburg Institute for Informatics and Automation (SPIIRAS),  
39, 14 Liniya, St.-Petersburg, 199178, Russia  
{ivkote, ulanov}@iiias.spb.su

**Abstract.** The paper considers the software simulation tool DDoSSim which has been developed for comprehensive investigation of Internet DDoS attacks and defense mechanisms. This tool can be characterized by three main peculiarities: agent-oriented approach to simulation, packet-based imitation of network security processes, and open library of different DDoS attacks and defense mechanisms. DDoSSim allows deeply investigating various attacks and defense methods and generating valuable recommendations on choosing the best defense. In the paper the agent-oriented approach suggested is considered. The taxonomy of input and output parameters for simulation is outlined. The main DDoSSim components are specified. One of the experiments on protection against DDoS attacks demonstrates some DDoSSim possibilities. We consider different phases of defense operations – learning, decision making and protection, including adaptation to the actions of malefactors.

**Keywords:** Security modeling and architecture, Security models for ambient intelligence environments, Infrastructure security, Security simulation, DDoS.

## 1 Introduction

The present theoretical investigations in information security of large-scale systems do not allow security experts to formalize adequately the antagonistic counteraction of network attacks and defense. Though the researchers can represent particular defense mechanisms, the understanding of security systems as holistic entities is a very difficult task. This understanding depends on many dynamical interactions between particular security processes and cyber-counteraction between antagonistic elements. It is especially right, taking into account the evolution of the Internet into decentralized distributed environment where a huge number of cooperating and antagonistic software agents exist and interact.

One of the very dangerous classes of malefactors' attacks is DDoS [16]. Distributed, dynamical and cooperative character of such attacks complicates attack detection and protection. Realizing effective DDoS defense system is a very complicated problem. Effective defense includes the mechanisms of attack prevention, attack detection, tracing the attack source and attack counteraction. Adequate protection can only be achieved by cooperation of different distributed components [17].

The main task of defense systems against DDoS is to accurately detect these attacks, quickly respond to them [26] and recognize the legitimate traffic that shares the attack signature and deliver it reliably to the victim [17]. *Traditional defense* include detection and reaction mechanisms. Different network characteristics are used for detection of malicious actions (for example, source IP address [21], traffic volume [8], and packet content [19], etc.). To detect abnormal network characteristics, many methods can be applied (for instance, statistical [12], cumulative sum, pattern matching, etc). As a rule, the reaction mechanisms include filtering [20], congestion control [14] and traceback [11]. But, as a result of several reasons (detection of DDoS attack is most accurate close to the victim, separation of legitimate is most successful close to the sources, etc.), adequate victim protection to constrain attack traffic can only be achieved by *cooperation of different distributed components* [16, 17]. There are a lot of architectures for distributed cooperative defense mechanisms [2, 4, 10, 17, 19, 26, 27, etc.]. For example, [2] proposes a model for an Active Security System, comprising components that actively cooperate in order to effectively react to a wide range of attacks. COSSACK [19] forms a multicast group of defense nodes which are deployed at source and victim networks. The SOS [10] uses a combination of secure overlay tunneling, routing via consistent hashing, and filtering. A collaborative DDoS defense system proposed in [27] consists of routers which act as gateways. The distributed defense system described in [26] protects web applications from DDoS attacks. The DefCOM system [17] uses a peer-to-peer network of cooperative defense nodes.

In our opinion, it is possible to answer soundly on the questions about defense against network attacks, including DDoS attacks, by modeling and simulation of present and new attacks and defense mechanisms. It is very important to use adequate modeling and simulation approach and powerful simulation environment which give a researcher an opportunity to comprehensively investigate various modes of attack and defense operation, insert new methods, analyze efficiency of defense (for example, false positives, false negatives; percent of normal traffic filtration), etc.

Our research goal is to suggest a common approach and simulation environment for investigation and elaboration of adequate defense methods against DDoS attacks which can produce well-grounded recommendations on the choice of defense mechanisms that are the most efficient in particular conditions. The rest of the paper is structured as follows. *Section 2* outlines the common approach for simulation. *Section 3* describes attack and defense mechanisms used. *Section 4* presents the taxonomy of input and output parameters for simulation. *Section 5* considers the software environment developed and analyses the issues of network topology selection. *Section 7* demonstrates the example of experiments provided. *Conclusion* outlines the main results and future work guidelines.

## 2 Simulation Approach

We try to use the *agent-oriented approach* to simulate security processes in the Internet. It supposes that the cybernetic counteraction is represented as the interaction of different teams of software agents [6, 24, 25]. The aggregated system behavior becomes apparent by means of the local interactions of particular agents in dynamic

environment that is defined by the model of computer network. We distinguish at least two agent teams: the team of agents-malefactors and the defense team. The agents from the same team collaborate to realize the threat or to defend the network.

It is assumed the competing agents gather information from different sources, operate with uncertain knowledge, forecast the intentions and actions of opponent, estimate the possible risks, try to deceive each other, and react on opponent's actions. The choice of behavior for each team depends on the chosen goal of functioning and is defined dynamically depending on the opposite team actions and the environment state.

The *mechanisms of agent coordination* are based on the three groups of procedures [24, 25]: acts consistency maintenance; agents' functionality monitoring and recovery; and communication selectivity support (to choose the most "useful" communication acts). The models of agent functioning are to foresee, what each agent knows, what task has to be solved and to which agent it must address its request to receive such information if it is outside of its competence. The messages of one agent are to be represented in such terms that are understandable by other agents.

It is supposed that agents are to be able to realize the mechanisms of self-adaptation. The team of agents-malefactors evolves with the aid of generation of new instances and types of attacks and attack scenarios to overcome the defense subsystem. The team of defense agents adapts to the actions of malefactors by changing the security policy, forming new instances of defense and security profiles.

The *conceptual model of agents' counteraction* includes: (1) Ontology of application domain containing application notions and relations between them; (2) protocols of teamwork (for team of malefactors and team of defense); (3) Models of individual, group and team behavior of agents; (4) Communication component for agent message exchange; (5) Models of environment – the computer network, including topological and functional components.

It is proposed to use *various models* to research the processes of cybernetic counteraction. The choice of specific models depends on the necessary simulation fidelity and scalability. For example, analytical models let imitate the global processes happening in Internet, but describe the processes only on an abstract level. Packet-level simulation gives the opportunities to imitate the proceeding processes with high fidelity. They represent the network attack and defense actions as the exchange of packets. The greatest fidelity is archived with the hardware testbeds, but it succeeds in simulating the sufficiently limited fragments of agents' interactions. The approach used in the paper is based on packet-level simulation with the use of tools for network processes imitation as basic level of simulation environment.

### 3 Attacks and Defense Mechanisms

**DDoS attacks agents** are divided into two classes: "daemon" and "master". Daemons are attack executors. Master coordinates them. On the preliminary stage daemons and master are deployed on available (already compromised) hosts. The important parameters are the quantity and "distribution" of agents. Then the phase of team establishing takes place. Daemons send to master the messages with information that they

are alive and ready to work. Master stores the information about team members and their status. The malefactor sets the mutual team goal – to start the DDoS attack in the given moment of time. Master receives the attack parameters. Its goal is to send it to all available daemons. Then daemons begin to act. Their local goal is to execute the master instruction. They start to send the attack packets to the given host in the given mode. Master examines daemons periodically to know that they are workable. Master controls the given attack mode by receiving the replies from daemons. When a daemon does not answer, master decides to change attack parameters. For example, it can send the commands to change the attack intensity to all or particular daemons. Daemons can execute the attack in several modes. This influences on the possibility of defense team to detect and block the attack and to trace and defeat the attack agents. The mode can be specified, for example, by the intensity of packet sending (packets per second) or (and) the method of IP address spoofing. The malefactor can stop the attack giving to master the command “stop the attack”. Master resends this command to daemons, and they stop the attack.

**Defense agents** are classified into the following classes: initial information processing (“sensor”); secondary information processing (“sampler”); attack detection (“detector”); filtering (“filter”); investigation (“investigator”).

In the initial moment the defense agents are deployed on the hosts corresponding to their roles: sensor and sampler – on the way of traffic to the defended host; detector – on any host of defended host subnet; filter – in the entrance to the defended host subnet; investigator – on any host outside of defended host subnet.

*Sensor* processes information of network packets and collects statistical traffic data for the defended host. Sensor can calculate the amount of traffic (bits per second – BPS) and determine the addresses of hosts that make the largest traffic. The functions of sensor can be fulfilled by *sampler*. Sampler processes the network packets and creates the model of normal functioning for the given network (in learning mode). Then in normal mode it analyses and compares the traffic with the model of normal traffic. It picks out the addresses of hosts that do not correspond to the model and sends them to detector. The examples of methods which can be realized by sampler are Hop counts Filtering (HCF) [9], Source IP address monitoring (SIPM) [22], Bit per Second (BPS), etc.

The *detector* local goal is to make a decision about the beginning of attack on the basis of sensor or (and) sampler data. Detector sends the list of attack addresses received from sensor or (and) sampler to filter and investigator. The *filter* local goal is to filter the traffic on the basis of detector data. The *investigator* goal is to trace and defeat the attack agents. After receiving a message from detector it examines the obtained IP addresses for the presence of attack agents and tries to defeat them.

#### 4 Taxonomy of Input and Output Parameters for Simulation

We differentiate the input parameters which specify DDoS attack and defense mechanisms for simulation.

The scheme of *DDoS attack parameters* is based on the attack taxonomy suggested in [15]. The following criteria were selected:

- *Victim type.* Application, host or network can be chosen. It is necessary to set victim IP address and port.
- *Attack type.* Brute-force (UDP/ICMP flood, smurf/fraggle, etc.) or semantic (TCP SYN, incorrect packets, hard requests).
- *Impact on the victim.* One can choose a disruptive attack (when all daemons attack simultaneously) or a degrading attack (when daemons join the attack one by one). It is easier to detect the attack in the first case.
- *Attack rate dynamics.* It can be constant or variable when the intensity changes in time. The function of changing attack packet rate is given to daemons. The change can be increasing (daemons send more and more packets) or fluctuating.
- *Agents' set permanency.* The set of agents can be persistent (all daemons participate in attack) or variable. In last case master can divide all daemons to several groups and each of them attacks alternately.
- *Possibility of exposure.* The attack can be discovered when it is possible to distinguish the attack packets. We distinguish non-filterable and filterable attacks. In non-filterable attack, the attack packets are formed to be indistinguishable from legitimate. In filterable attack, the attack packets can be discovered by field values, size, exploited protocol, etc.
- *Source addresses validity.* Attacker can use the valid (real) or spoofed source address sending the attack packets. This address can be routable or non-routable. The method of spoofing may be as follows: (1) Without spoofing ("no") – the real address of host (where daemon is deployed) is used; (2) "Constant" – an address is randomly chosen, then it is used for sending the attack packets; (3) "Random" – with every new attack packet a new address from the given range of addresses is randomly chosen. This range does not intersect with the range of addresses used in the given network; (4) "Random real" – with every new attack packet a new address from the given range of addresses is randomly chosen. This range is in the range of addresses used in the given network.
- *Degree of automation.* Attack can proceed automatically after setting the parameters or by the malefactor control. In such a case he (she) can interfere and change one of parameters on all phases of attack. The communication mechanisms between daemons and master can be direct (master knows the addresses of all daemons) or indirect (agents communicate via a server).

The scheme of *DDoS defense parameters* is built on the basis of classification proposed by authors. The criteria selected are as follows:

- *Deployment location:* source, intermediate or defended subnets.
- *Mechanism of cooperation.* The mechanism of particular components operation can be centralized or decentralized. In the last case the defense components are autonomous and can combine their efforts.
- *Covered defense stages.* The stages (mechanisms) the defense method can implement are as follows: (1) attack prevention; (2) attack detection; (3) attack source detection; (4) attack counteraction.
- *Attack detection technique.* There are two types of detection: misuse and anomaly. One chooses one particular detection method or the set of methods.

- *Attack source detection technique.* Attack source detection (or “traceback”) can be realized by packet signatures, packet marking, generation of auxiliary packets, etc.
- *Attack prevention/counteraction technique.* One can use filtering (of packets or flows), resource management (differentiation, change of quantity, roaming) and authentication.
- *Technique for model data gathering.* Data can be generated by learning or be obtained from external sources.
- *Determination of deviation from model data.* One can use thresholds, rules (for packets and connections), determining fluctuation in probabilistic traffic parameters, and data mining (depending on the kind of defense mechanism).

The *output parameters* used to estimate the defense mechanisms are as follows: List of detectable attacks; Time of attack detection (from the start of attack); Time of attack reaction (time from detection to counteraction); Percent of false positives; Percent of false negatives; Percent of normal traffic filtration; Computational complexity (quantity of computational resources used), etc.

## 5 Simulation Environment

The simulation environment DDoSSim architecture consists of the following components (figure 1): OMNeT++ Framework, INET Framework, Multi-agent & DDoS Framework.

Multi-agent simulation is implemented in Multi-agent Framework that uses the library of DDoS attack and defense mechanisms called DoS Framework. INET Framework is used to simulate the IP nodes. It is an OMNeT++ model itself.

*OMNeT++ Framework* [18] is a discrete event simulator. Simulation models are composed of hierarchically nested modules that interact due to message passing (figure 1, OMNeT++ Framework: simulation model and component library). INET Framework and Multi-agent DDoS Framework are the OMNeT++ models. The exchange of messages between modules happens due to channels (modules are connected with them by the gates) or directly by gates. A gate can be incoming or outgoing to receive or to send messages accordingly. Channel has the following properties: propagation delay, bit error rate and transmission data rate.

*OMNeT++ INET Framework* is the OMNeT++ modular simulation suite with a realistic simulation of the Internet nodes and protocols. The highest IP simulation abstraction level is the network itself, consists of IP nodes. IP node can represent router or host. IP node in INET Framework corresponds to the computer representation of Internet Protocol (figure 1, INET Framework). The modules of IP node are organized in such a way like operating system process IP datagram. The module that is responsible for network layer (implementing IP processing) and the “network interface” modules are mandatory. Additionally one can plug the modules that implement higher layer protocols: transport (UDP, TCP, including TCP Sockets; routing: MPLS, LDP, RSVP, OSPF-TE) and application (HTTP, Telnet).

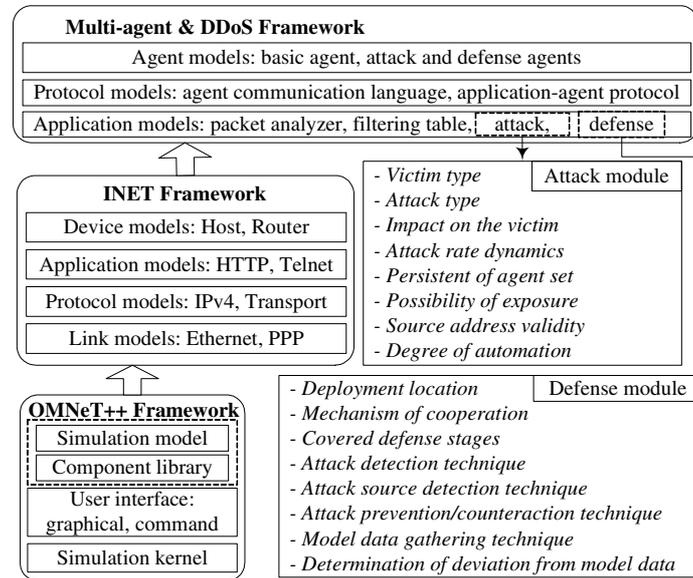


Fig. 1. DDoSSim Simulation environment architecture

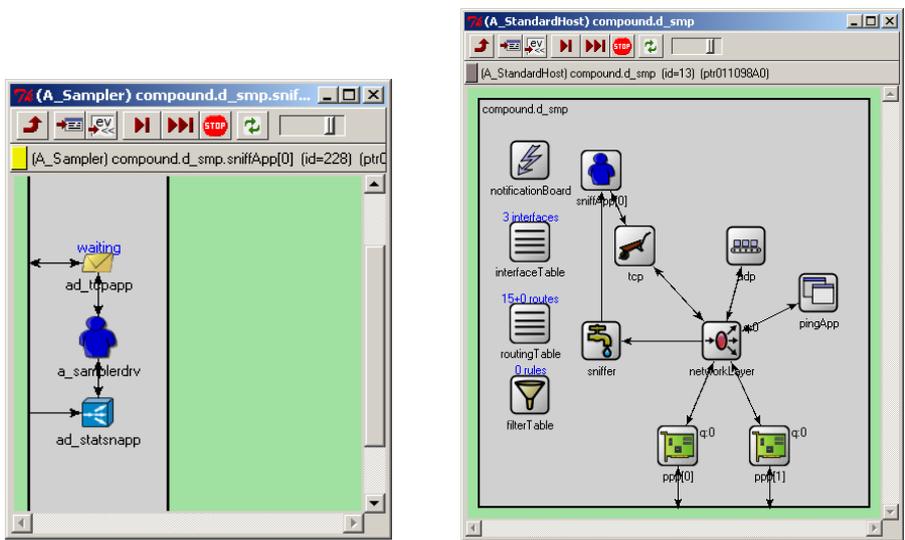
*Multi-agent & DDoS Framework* is the INET Framework modular suite aimed to simulate the DDoS attack and defense mechanisms on the basis of agent team counteraction (figure 1, Multi-agent DDoS Framework). One can distinguish between DDoS Framework and Agent Framework architecturally.

*DDoS Framework* suite consists of DDoS attack and defense modules (figure 1, Attack module, Defense module) and the modules that expand IP node from INET: the filtering table and the packet analyzer. Attack and defense modules are the applications and are deployed on the network layer of IP node. There were implemented different DDoS attacks and defense mechanisms, for example, Hop-count Filtering (HCF), Source IP address monitoring (SIPM), BPS, etc. To set the DDoS attack conditions it is necessary to define the corresponding parameters, including victim type (host), attack rate dynamics (function of attack packets sending rate), spoofing technique (no spoofing, random, subnet), etc. Also one need to set up the defense parameters, including deployment location (defended, intermediate, source subnet), detection technique, model data gathering technique and its parameters (time interval and time shift of data collection), etc.

*Agent Framework* consists of modules representing agents which are implemented as applications. There were used the elements of abstract FIPA architecture [7] during agent modules design and implementation. Agent communication language is implemented for agent interactions. The message passing happens above TCP protocol (transport layer). Agent directory is mandatory only for agents that coordinate other agents in teams. Agent can control other modules (including DDoS Framework modules) due to messages.

Agents are deployed on hosts in the simulation environment. Their installation is fulfilled by connecting to the modules serving transport and network layers of protocol

stack simulated in OMNeT++ INET Framework. The *generalized representation of agent* “sampler” structure is depicted in figure 2. Sampler contains the transport layer (depicted as a message), needed to communicate with other agents, network layer (depicted as a blue cube) to collect traffic data and agent kernel (depicted as a shape of human image). The agent kernel contains communication language, knowledge base and message handlers from the neighbor modules. The representation of sampler *deployment into simulation environment* is depicted in figure 3. One can see that the agent is plugged into host through the “tcp” module implementing TCP protocol. Agent is also connected with the “sniffer” module used to analyze network packets.



**Fig. 2.** General structure of agent “sampler” **Fig. 3.** Deployment of agent “sampler” into the environment

The example of *multi-window user interface of the simulation environment* is depicted in figure 4. At the basic window of visualization (figure 4, at upper right), a simulated computer network is displayed.

The window for simulation management (at the bottom right of figure 4) allows looking through and changing simulation parameters. It is important that you can see the events which are valuable for understanding attack and defense mechanisms on a time scale. The time scale is depicted above windows with the events description.

Corresponding status windows show the current status of agent teams (see the defense team status window at the upper left of figure 4). It is possible to open different windows which characterize functioning (the statistical data) of particular hosts, protocols and agents (see these windows at the bottom left of figure 4).

The example of hierarchy of simulated objects is represented in figure 5 (from left to right there are showed the nested objects “network”, “host”, “agent”). During investigation one can move from one hierarchy level to another and analyze functioning parameters of various objects.

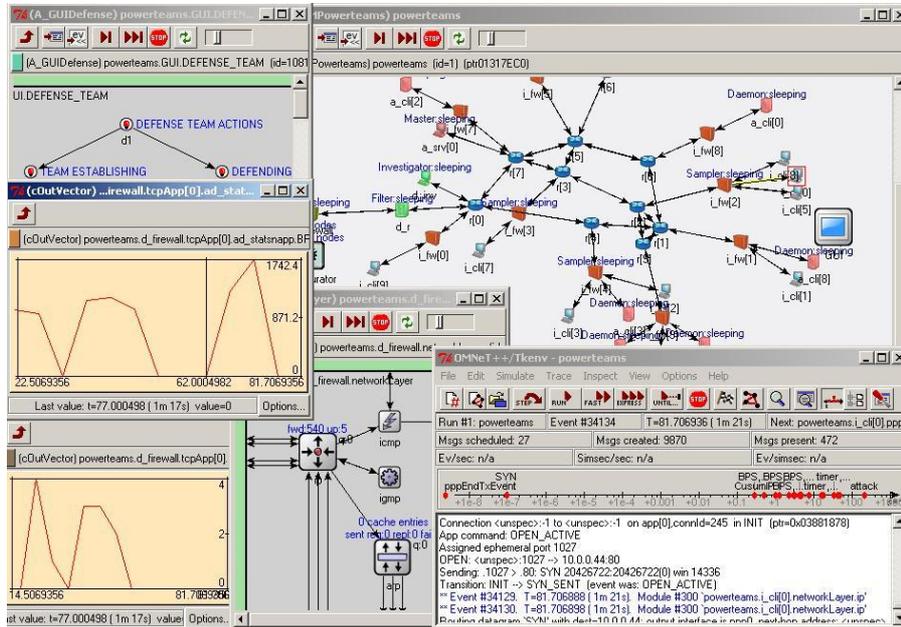


Fig. 4. Common representation of the simulation environment

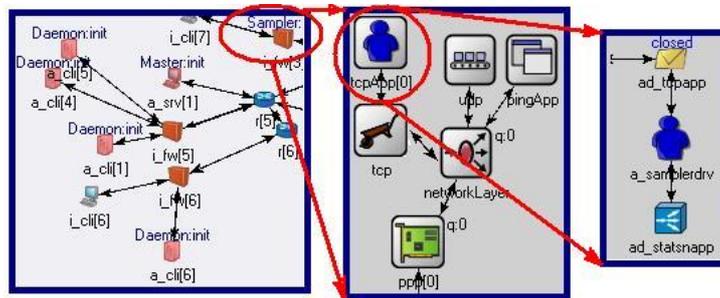


Fig. 5. Example of object hierarchy: “network” → “host” → “agent”

At the basic window of visualization (figure 6), a simulated computer network is displayed. The network represents a set of hosts and channels. Hosts can fulfill different functionality depending on their parameters or a set of internal modules. The routers are depicted with the sign “”. Attack agents are deployed on the hosts marked with red color. Defense agents are located on the hosts marked with green color. Above the colored hosts there are strings indicating the corresponding state of deployed agents. The other hosts are standard hosts that generate normal traffic.

Each network for simulation consists of three types of sub-networks: (1) the subnet of defense where the defense team is deployed; (2) the intermediate subnet where the standard hosts are deployed. They produce the generic traffic including the traffic to defended host; (3) the subnet of attack where the attack team is deployed.

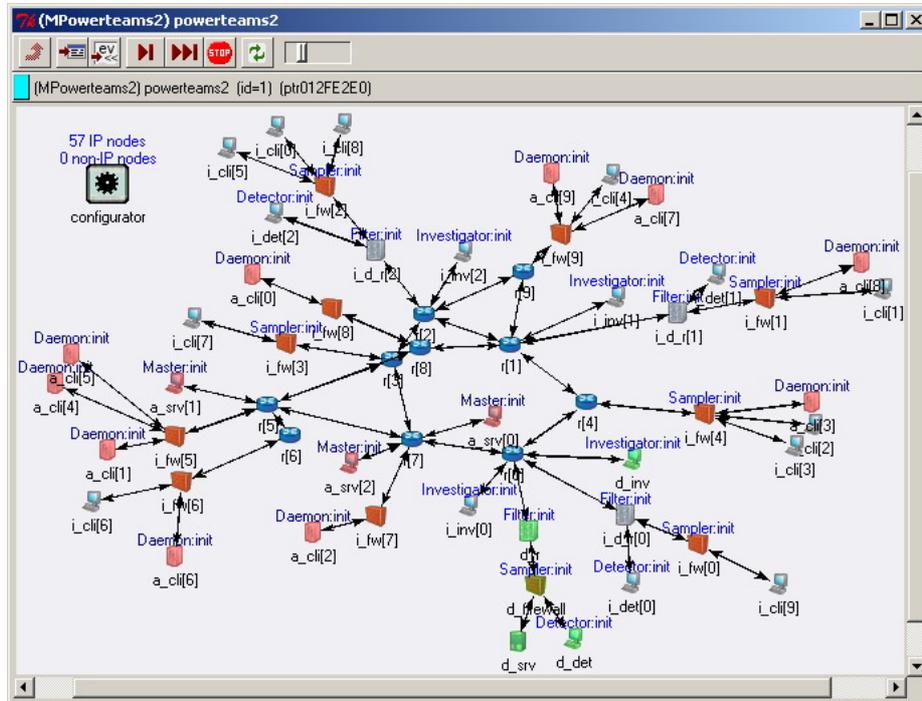


Fig. 6. Example of computer network for simulation

The subnet of defense as a rule includes at least five hosts. The following agents are deployed on the first four hosts: detector, sampler, filter and investigator. The web-server which is under defense is deployed on the fifth host. The agents and the web-server are the applications installed on corresponding hosts. The IP addresses are being set automatically. It is necessary to set the other application parameters. Web-server is deployed on the host d\_srv. The interaction port and the answer delay must be set. Detector is deployed on the host d\_det. The following parameters are used for detector: the defended host IP address, the port for team interaction, the interval for sensor inquiry, and the maximum allowed data-rate to server (BPS, bit per second). Sampler is placed on the host d\_firewall (on the entrance to the server subnet). Filter is installed on the host d\_r (router). Investigator is deployed on the host d\_inv. For each of the last three agents, the private port, the IP address of detector and the port for team interaction must be determined.

The intermediate subnet includes N hosts i\_cli[...] with generic clients. They are connected by the router i\_r. The number of hosts N is the simulation parameter which can be set. The following parameters of clients must be specified: IP-address and port of server, the time of work start, the quantity and size of requests while connecting to server, the size of reply and the time of reply preparation, the idle interval.

The subnet of attack consists of M hosts i\_cli[...] with daemons and one host with master. The number of hosts M must be set. Master has the following parameters: port for team interaction, IP-address and port of attack target, the time of start of attack

and its rate (measured in packets per second). Daemon has the following parameters: the port, masters' IP-address and port for team interaction.

To simulate the Internet processes (including DDoS defense and attack mechanisms), we needed the models of single hosts and topology as the representation of the way these hosts are connected. In relevant publications (e.g. [1], [23]) the Internet is represented as a graph built due some analytical dependencies. One of the main parameters to build the correct graph is node degree  $k$ . It is the amount of nodes with which the given node is connected. The average node degree is defined by the following formulae:  $k = 2m/n$ , where  $m$  is the amount of connections, and  $n$  is the amount of nodes in the network.

The network topology which is similar to Internet can be built on the basis of node degree. The function that can determine  $k$  for every node in network is needed. [13] summarizes the data on investigating this function. The probability density function (PDF) of node degree is built upon the data from the distributed sensors ("skitters"), BGP tables and WHOIS [13]. PDF of  $k$  for the Internet is similar to the function  $f(k) = ck^{-\gamma}$ , and the values of  $k$  are bounded in the following way [5]:

$$k_{\min} \leq k \leq k^{1/(\gamma-1)}.$$

The graph that represents the network topology is built in the following way [3]. There is chosen the amount of nodes  $n$ . It is generated the random value  $k_i$  on the basis of distribution  $f(k)$  for every node (the sum of  $k_i$  must be even). Then every node  $i$  from the set is connected with the other  $k_i$  randomly chosen node. There are the other ways to build the random graph. The generation due to clustering method and joint degree distribution are more precise [13]. The basic network that is used for simulations in the developed environment is built in compliance with described algorithm and PDF (\*). The value  $\gamma = 2.25$  is borrowed from [13]. On the basis of experimental data the minimum node degree is 2.

## 6 Simulation Scenario Examples

The attack parameters used in the experiments represented in the paper are as follows (see section 4): Victim type – host (server that provides some service); Attack type – brute-force; Impact on the victim – disruptive; Attack rate dynamics – constant, variable; Agents' set permanency – constant, variable; Possibility of exposure – discoverable filterable attack; Source addresses validity – valid (real), spoofed: random, subnet; Degree of automation – semi-automatic with direct communication.

In the experiments considered in the paper the following defense parameters were used (see section 4): Deployment location – intermediate, defended subnets; Mechanism of cooperation – centralized; Covered defense stages – attack prevention, attack detection, attack source detection, attack counteraction; Attack detection technique – anomaly detection (Hop-count Filtering (HCF), Source IP address monitoring (SIPM), Bit per Second (BPS)); Attack source detection technique – can detect when source address is not spoofed; Attack prevention technique – packet filtering; Technique for gathering of model data – learning; Determination of deviation from model

data: thresholds (HCF, BPS), determination of fluctuation in probabilistic traffic parameter (SIPM).

**Learning mode.** The main task of learning mode is to create the model of generic traffic for the given network. The clients send the requests to the server and it replies. At this time sampler analyses requests and uses them to form the models of normal traffic and other parameters. During the learning it is possible to watch the change of traffic models (see figures 7-11).

Figure 7 represents the list of hosts that sent requests to server and hops to them after 300 seconds of learning and the time of last request. As mentioned above the hop count is calculated on the basis of TTL packet field.

Figure 8 depicts the change of new addresses amount for sampler during first 300 seconds of learning. One can see that in the beginning when clients requested server at the first time there were many new addresses (the maximum is 6 addresses, the time interval is 10 seconds, and the shift is 3 seconds). The last unknown address appeared in the region of 100 first seconds. At least, when all clients have requested the server there were no new addresses.

Figure 9 shows the list of clients requested the server and considered as legitimate after first 300 seconds of learning. One can see here that in the interval between 0 and 50 seconds there were many new addresses.

Figure 10 represents the graph of change of maximum BPS (for interval 10 seconds and shift 3 seconds) after 300 seconds from the beginning of learning. The maximum value was 1742.4 bit/s and was recorded in the area of 100 seconds. One can see also the values of BPS for clients that requested server in the current time interval.

Figure 11 depicts the values of transmitted bits for every client that requested server in the interval of 10 seconds.

**Decision making and acting.** Simulation scenario is realized on the same configuration as was used during learning. The only difference is that the attack team is engaged. Attack team initial parameters are as follows: `target_ip="d_srv"` (target of attack is server `d_srv`); `target_port="2001"` (target port); `t_ddos=300` (time of attack start); `attack_rate=5` (intensity of attack in packets per second); `ip_spoofing="no"` (no IP spoofing is used).

Figure 12 represents the graphs of channel throughput (bits/s to seconds) on the entrance to the defended network before (dashed line) and after (firm line) filter.

After modeling start the clients begin to send requests to the server and it replies. This is the way the generation of generic network traffic takes place (figure 12, interval 0 – 300 seconds). The formation of defense team occurs after some time from start. Investigator, sampler and filter connect to detector and send it the messages that they are alive and ready to work. Detector stores this information. The attack team is formed in the same way. Daemons connect to master and report their status. After establishing the defense team begins to function. Sampler collects traffic data and compares it with the model data acquired during learning mode. The addresses that are the source of anomalies are sent to detector every  $n$  seconds (in this scenario  $n=60$ ). Detector makes the decision about the attack and sends to filter and investigator the addresses of suspicious hosts.

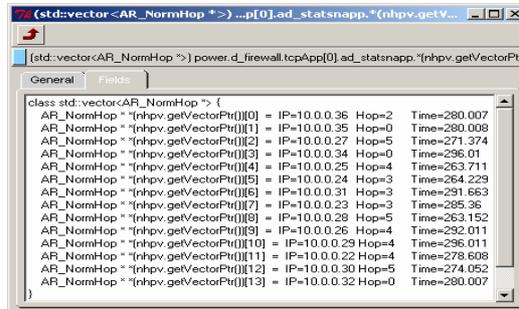


Fig. 7. List of hosts that sent requests to server and hops to them after 300 sec of learning

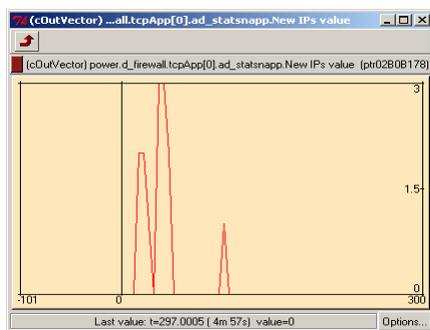


Fig. 8. Change of new IP addresses amount

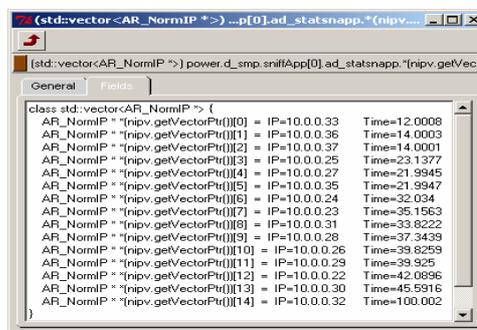


Fig. 9. List of clients requested server and considered as legitimate after 300 sec of learning

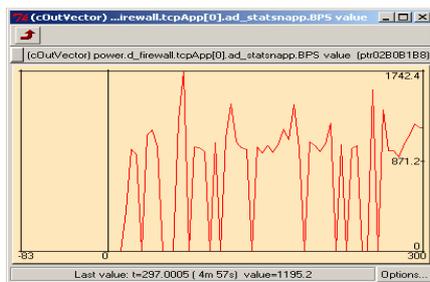


Fig. 10. Change of BPS parameter

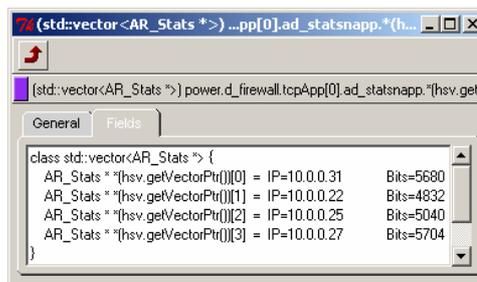


Fig. 11. Values of transmitted bits

After 300 seconds from simulation start the attack team begins attack actions. Master examines all daemons that it knows. Then it sends the command of attack to all workable daemons. This command includes address and port of attack target, intensity (distributed among daemons) and the method IP spoofing. In this case they are: target – d\_srv, port – 2001, intensity of attack for every daemon (calculated as intensity divided by the number of daemons)  $5/10=0.5$ , spoofing “no” (no IP spoofing). When

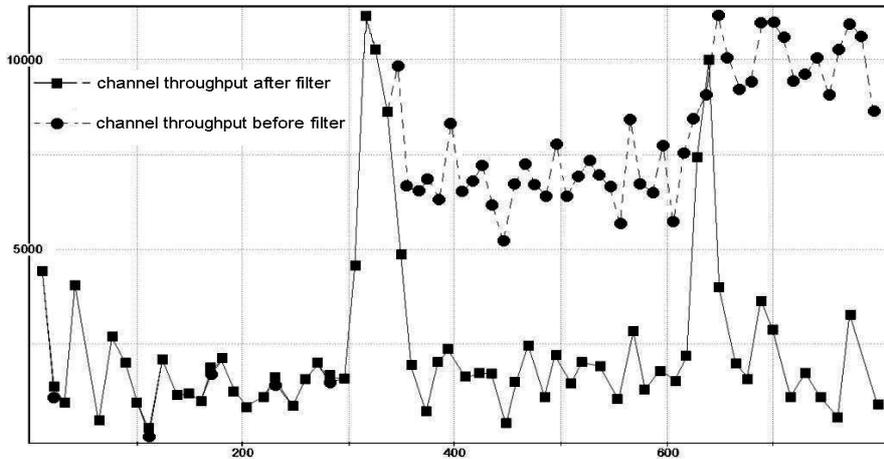


Fig. 12. Graphs of channel throughput

daemons receive the command they begin to send the attack packets (figure 12, timestamp 300 seconds).

After a while, sampler determines the suspicious hosts with the use of BPS method. The BPS parameter of these hosts exceeds normal value. Detector receives the addresses of these hosts from sampler and sends them to filter and investigator. Filter sets the filtering rules and the packets from the given hosts begin being dropped (figure 12, timestamps 400 – 600 seconds, firm graph).

Investigator tries to inspect the given hosts and to defeat the attack agents deployed there. It succeeds in defeating of four daemons. The string “defeated” appears above the defeated agent in the window of network structure. However the other daemons continue the attack (figure 12, after 400 seconds, dashed graph).

Master examines daemons next time 600 seconds after simulation has started. It does not succeed to connect with all daemons as some of them were defeated by investigator. Master makes the decision to redistribute the intensity of attack to keep the overall intensity on the given level. Also it decides to change the method of IP spoofing to complicate the detection and defeating of attack agents by defense team. Master sends to alive daemons the command: target – d\_srv, target port – 2001, intensity –  $5/(10-4)=0.83$ , IP spoofing method – “random”. When daemons receive the command they continue to send the attack packets having applied the new parameters (figure 12, timestamp 600 seconds).

Detector sees that the input channel throughput has noticeably lowered since the traffic from attack team has raised (figure 12, after 600 seconds). Detector does not receive the anomaly report from sampler though. This is because the method BPS used by sampler does not work fine when attacker changes the sender address in every packet. That is the reason detector fails to confront some address with the big traffic. Therefore detector decides to apply another DDoS defense method – SIPM. The large amount of new IP addresses for sampler will lead to attack detection and dropping malicious packets. This method however does not allow tracing the source

of attack directly, and investigator will fail to defeat attack agents. But the attack packets will be filtered and the traffic in the subnet of defended host will return to normal state dropped (figure 12, timestamps 400 – 600 seconds, firm graph).

The following *effectiveness and efficiency parameters of different defense mechanisms* were studied during experiments: rate of dropped legitimate traffic (false positive rate); rate of admitted attack traffic (false positive rate); attack reaction time. These parameters were investigated in dependence on the following *input parameters*: network configuration (the amount of legitimate clients); attack intensity; IP address spoofing technique used in attack; internal parameters of defense mechanisms and their combinations; quantity and distribution of defense teams, etc.

## 7 Conclusion

The main results of the work we described in the paper consist in developing an approach to agent-based simulation of defense mechanisms against attacks and implementing the software environment DDoSSim intended for simulation of DDoS attacks and defense. The goal of the paper is not to present an investigation of new defense methods, but to show the possibilities of the simulation tool developed. One of the features of this tool is the possibility to insert new attack and defense methods and investigate them. The environment developed is written in C++ and OMNeT++. It allows imitating a wide spectrum of real life DDoS attacks and defense mechanisms.

Various experiments with this environment have been fulfilled. These experiments include the investigation of attack scenarios and protection mechanisms for the networks with different structures and security policies. One of the scenarios was demonstrated in the paper. Future work is connected with building more powerful simulation environment based on large library of DDoS attack and defense mechanisms, investigating new defense mechanisms, and conducting experiments to both evaluate computer network security of large-scale network security solutions and analyze the efficiency and effectiveness of different security policies against various attacks. The special attention will be given to cooperative defense mechanisms that are based on the deployment of defense components in various Internet subnets.

**Acknowledgments.** The research is supported by grant of Russian Foundation of Basic Research (№ 04-01-00167), Department for Informational Technologies and Computation Systems of the Russian Academy of Sciences (contract №3.2/03), Russian Science Support Foundation and by the EC as part of the POSITIF project (contract IST-2002-002314).

## References

1. CAIDA. <http://www.caida.org/tools/>
2. Canonico, R., Cotroneo, D., Peluso, L., Romano, S.P., Ventre, G.: Programming routers to improve network security. Proceedings of the OPENSIG 2001 Workshop Next Generation Network Programming (2001)
3. Catanzaro, M., Boguñá, M., Pastor-Satorras, R.: Generation of uncorrelated random scale-free networks. Physical Review E 71, 027103 (2005)

4. Chen, S., Song, Q.: Perimeter-Based Defense against High Bandwidth DDoS Attacks. *IEEE Transactions on Parallel and Distributed Systems*, Vol.16, No.7 (2005)
5. Dorogovtsev, S.N., Mendes, J.F.F.: The shortest path to complex networks. <http://arxiv.org/abs/cond-mat/0404593> (2004)
6. Fan, X., Yen, J.: Modeling and Simulating Human Teamwork Behaviors Using Intelligent Agents. *Journal of Physics of Life Reviews*, Vol. 1, No.3 (2004)
7. FIPA. <http://www.fipa.org>
8. Gil, T.M., Poletto, M.: MULTOPS: a data-structure for bandwidth attack detection. *Proceedings of 10th Usenix Security Symposium* (2001)
9. Jin, C., Wang, H, Shin, K.G.: Hop-count filtering: An effective defense against spoofed DDoS traffic. *Proceedings of the 10th ACM Conference on Computer and Communications Security* (2003)
10. Keromytis, A.D., Misra, V., Rubenstein, D.: SOS: An architecture for mitigating DDoS attacks. *Journal on Selected Areas in Communications*, Vol. 21 (2003)
11. Kuznetsov, V., Simkin, A., Sandström, H.: An evaluation of different ip traceback approaches. *Proceeding of the 4th International Conference on Information and Communications Security* (2002)
12. Li, M., Chi, C.H., Zhao, W., Jia, W.J., Long, D.Y.: Decision Analysis of Statistically Detecting Distributed Denial-of-Service Flooding Attacks. *Int. J. Information Technology and Decision Making*, Vol.2, No.3 (2003)
13. Mahadevan, P., Krioukov, D., Fomenkov, M., Huffaker, B., Dimitropoulos, X., Claffy, K., Vahdat, A.: Lessons from Three Views of the Internet Topology: Technical Report. Cooperative Association for Internet Data Analysis (CAIDA) (2005)
14. Mahajan, R., Bellovin, S.M., Floyd, S.: Controlling High Bandwidth Aggregates in the Network. *Computer Communications Review*, Vol.32, No.3 (2002)
15. Mirkovic, J., Martin, J., Reiher, P.: A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms, Technical report #020018. University of California, Los Angeles (2002).
16. Mirkovic, J., Dietrich, S., Dittrich, D., Reiher, P.: *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR (2004)
17. Mirkovic, J., Robinson, M., Reiher, P., Oikonomou, G.: Distributed Defense Against DDOS Attacks. University of Delaware. Technical Report CIS-TR-2005-02 (2005)
18. OMNeT++ homepage. <http://www.omnetpp.org/>
19. Papadopoulos, C., Lindell, R., Mehringer, I., Hussain, A., Govindan, R.: Cossack: Coordinated suppression of simultaneous attacks. *Proceedings of DISCEX III* (2003)
20. Park, K., Lee, H.: On the Effectiveness of Route-based Packet Filtering For Distributed DoS Attack Prevention in Power-law Internet. *Proceedings ACM SIGCOMM* (2001)
21. Peng, T., Christopher, L., Kotagiri, R.: Protection from Distributed Denial of Service Attack Using History-based IP Filtering. *IEEE International Conference on Communications* (2003)
22. Peng, T., Leckie, C., Kotagiri, R.: Proactively Detecting DDoS Attack Using Source IP Address Monitoring, *Networking 2004*, Athens, Greece (2004)
23. Route-Views Bibliography. <http://www.routeviews.org/papers/>
24. Tambe, M.: Towards flexible teamwork. *Journal of AI Research*, Vol.7 (1997)
25. Tambe, M., Pynadath, D.V.: Towards Heterogeneous Agent Teams. *Lecture Notes in Artificial Intelligence*, Vol.2086 (2001)
26. Xiang, Y., Zhou, W.: An Active Distributed Defense System to Protect Web Applications from DDoS Attacks. *The Sixth International Conference on Information Integration and Web Based Application & Services* (2004)
27. Xuan, D., Bettati, R., Zhao, W.: A gateway-based defense system for distributed dos attacks in high-speed networks. *IEEE Transactions on Systems, Man, and Cybernetics* (2002)