Multiagent modeling and simulation of agents' competition for network resources availability

Igor Kotenko[1] and Alexander Ulanov[2]

St.-Petersburg Institute for Informatics and Automation, Russia
39, 14th Liniya, St. Petersburg, 199178, Russia
[1] ivkote@iias.spb.su; [2] ulanov@iias.spb.su

Abstract. The paper considers an approach to modeling and simulation of competition in Internet between the antagonistic teams of software agents for network resources availability. We differentiate two teams: (1) the attack team realizing Distributed Denial of Service (DDoS) attacks and (2) the defense team protecting against these attacks. Teamwork based approach for modeling and simulation of agents' competition is considered. The structure and operation of agents' teams, their ontologies, main classes and roles of agents are defined. We describe in detail the case-study which demonstrates main ideas of the approach suggested.

1 Introduction

Permanently magnified variety and complexity of cyber-attacks and gravity of their consequences highlights urgent necessity for strong protection mechanisms of computer systems. Especially it is fair in connection with integration of computer systems on the basis of the Internet, not having state boundaries, centralized control and uniform security policy.

Unfortunately, the existing theoretical base for network security in large-scale systems does not correspond to the indicated tendencies. We think the majority of problems is caused by immaturity of logical foundations for construction of integrated adaptive security systems [4]. To our opinion, it is stipulated mainly by insufficient attention to fundamental works, which, on the one hand, consider network security as a complex task of organizational and technical competition between security systems and malefactors' systems [6], and, on the other hand, are based on *exploratory modeling and simulation* of indicated processes.

The issues of modeling and simulation of network security have been actively researched for more than thirty years. The various formal and informal models of particular protection mechanisms are developed, but practically there are not enough works formalizing complex antagonistic character of network security. Understanding of network security as uniform holistic system is extremely hampered. It depends on great many interactions between different cyber warfare processes and is determined by dynamic character of these processes and different components of computer systems. Especially it is fair in conditions of the Internet evolution to a free decentralized distributed environment in which a huge number of cooperating and

antagonistic software components (agents) interchange among themselves and with people by large information contents and services. Modeling and simulation of these aspects is supposed to put as a basis of our research. This will allow developing an integrated approach to construction of network security systems which can operate in aggressive antagonistic environment.

One of the most harmful classes of attacks aiming at destruction of network resources availability is Denial of Service (DoS). The purpose of DoS is isolation of a victim host, i.e. creation of a situation in which a remote host can not communicate with external world. The basic feature of Distributed Denial of Service (DDoS) attacks is coordinated use of enormous remote hosts-zombies for generation of ill-intentioned traffic [21, 22].

The purpose of our research is twofold. Firstly, we try to develop formal basis for adaptive co-evolving agent-based modeling and simulation of antagonistic agents' teams. Secondly, we aim to suggest an approach for modeling and simulation of agents' competition in the Internet for network resources availability. In this paper we investigate our approach on an example of implementing DDoS attacks and protecting against them. The rest of the paper is structured as follows. *Section 2* reviews relevant works and outlines suggested common approach for modeling and simulation of antagonistic agents' team competition in the Internet. *Section 3* describes the issues of modeling and simulation of attack agents' team. *Section 4* presents the aspects of modeling and simulation of defense agents' team. *Section 5* outlines architecture and main user interfaces for software prototype developed. *Conclusion* outlines the main results of the paper and future work directions.

2 Teamwork based Approach and Related Work

The agents' team realizes teamwork, if the team members fulfill joint operations for reaching the common long-time goal in a dynamic external environment at presence of noise and counteraction of opponents. The teamwork is something greater, than simply coordinated set of personal actions of individual agents. It is accepted to speak, that in teamwork the agents collaborate. The collaboration is a special sort of a coordinated activity of the agents, in which they jointly solve some task or fulfill some activity for reaching a common goal. The main problem at organization of the agents' teamwork is how it is possible to provide actions of the agents as united team in a situation, when each agent realize own intentions by personal operations executed in parallel or sequentially with operations of other agents [5, 12, 15, 29, 30, 34].

Several models and software/hardware products have been developed for producing agents' teamwork **[**5]. One of the approaches, known as the joint intentions theory, is offered in [3]. It states a common framework determining a team behavior and an interaction of the team members. The more formalized approach, known as the shared plans theory, is offered in [10, 11]. In [29, 30] the key ideas of both approaches are generalized and used at creation of software toolkit for development of applications in the field of teamwork of the agents. The general intentions of agents are determined in a hierarchical reactive plan. This plan describes actions of the team as well as the actions of particular agents. The coordinated tasks are carried out due to

installation of constraints on agents' roles. GRATE* [12] is an implementation of teamwork using the Joint Responsibility model. This model includes concepts of common goals and instructions (recipes). The individual commitments determine how an agent should operate in a context of teamwork. OAA (Open Agent Architecture) [20] uses a blackboard-based framework that allows individual agents to communicate by means of goals posted on blackboards controlled by facilitator agents. CAST (Collaborative Agents for Simulating Teamwork) [34] supports teamwork using a shared mental model. The mental model includes team processes, team structures and the capability of each teammate. In RETSINA-MAS [7], agents have own copy of a common partial plan. Each agent estimates its opportunities to the requirements of the team goal. In " Robocup Soccer " [28, 15], agents have common knowledge operating their cooperative behavior. COGNET/BATON [35] is a system for simulation of teamwork of people with use of intelligent agents. Team-Soar [13] is a model implemented for testing a theory of team decision making.

In our approach it is offered that the agents' teamwork is organized by the group (team) plan of the agents' actions. In result, a team has a mechanism of decision-making about who will execute particular operations. As in the joint intention theory, the basic elements, allowing the agents' team to fulfill a common task, are common (group) intentions, but its structuring is carried out in the same way as the plans are structured in the shared plans theory [10]. The common (group, individual) intention and commitment are associated with each node of a general hierarchical plan. These intention and commitment manage execution of a general plan, providing necessary flexibility. During functioning each agent should possess the group beliefs concerning other team-mates. For achievement of the common beliefs at formation and disbandment of the common intentions the agents should communicate. All agents' communications are managed by means of common commitments built in the common intentions. For this purpose it is supposed to use the special mechanism for reasoning of agents on communications. Besides it is supposed, that agents communicate only when there can be an inconsistency of their actions. It is important for reaction to unexpected changes of network environment, redistributing roles of the agents which failed or unable to execute the general plan, and also at occurrence of not planned actions. The mechanisms of the agents' interaction and coordination are based on three groups of procedures [29]: (1) Coordination of the agents' actions (for implementation of the coordinated initialization and termination of the common scenario actions); (2) Monitoring and restoring the agents' functionality; (3) Communication selectivity support (for choice of the most "useful" communications).

The specification of the plan hierarchy is carried out for each role. The following elements of the plan should be described: initial conditions, when the plan is offered for fulfillment; conditions for finishing the plan execution (these conditions can be as follows: plan is fulfilled, plan is impracticable or plan is irrelevant); actions fulfilled at the team level as a part of the common plan. For the group plans it is necessary to express joint activity. To cope with the information heterogeneity and distribution of intrusion sources and agents used we apply ontology-based approach and special protocols for specification of shared consistent terminology. The ontology of network security problem and application domain is specified on the basis DAML+OIL.

The suggested technology for creation of the malefactors-agents' team (that is fair for other subject domains) consists in realization of the following chain of stages [16]:

(1) Formation of the subject domain ontology; (2) Determination of the agents' team structure; (3) Determination of agents' interaction-and-coordination mechanisms (including roles and scenarios for roles exchanges); (4) Specification of agents' plans; (5) Assignment of roles and allocation of plans between agents; (6) State-machine based implementation of teamwork.

The agents' team structure is described in terms of a hierarchy of group and individual roles. Leaves of the hierarchy correspond to the roles of individual agents, but intermediate nodes - to group roles. One agent can execute a set of roles. Agents can exchange roles in progress of plan execution. Agents' coordination is carried out by message exchange. As the agents' teams operate in antagonistic environment agents can fail. The lost functionalities are restored by redistributing the roles of failed agents between other agents and (or) cloning new agents.

Agent-based modeling and simulation of network security in the Internet assumes that agents' competition is represented as a large collection of semi-autonomous interacting agents. The aggregate system behavior emerges from evolving local interactions of agents in a dynamically changing environment specified by computer network model. We assume to select at least two antagonistic agents' teams effecting on computer network as interconnected set of resources: (1) Attack system is a team of attack agents (automatically generating DDoS attacks); (2) Defense system is a team of security agents (for intrusion protection, data sensing and information fusion, intrusion detection, and incident response).

The main task of defense systems against DDoS is to accurately detect these attacks and quickly respond to them [31]. It is equally important to recognize the legitimate traffic that shares the attack signature and deliver it reliably to the victim [23]. Traditional defense include detection and reaction mechanisms [33]. Different network characteristics are used for detection of malicious actions (for example, the source IP address [27], the traffic volume [8], and the packet content [26]). To detect the abnormal network characteristics, many methods can be applied (for instance, statistical [18], cumulative sum, pattern matching, etc). As a rule, the reaction mechanisms include filtering [25], congestion control [19] and traceback [17].

Let us consider some papers that provide defense from DDoS attacks by cooperative actions of different components. The paper [1] proposes a model for an Active Security System, comprising a number of components that actively cooperate in order to effectively react to a wide range of attacks. COSSACK [26] forms a multicast group of defense nodes which are deployed at source and victim networks. Each defense node can detect the attack and alert the other nodes. This system combines multicast communications, traditional intrusion detection components, network topology, vulnerability information, and blind detection techniques. In [14] an architecture called Secure Overlay Services (SOS) is described. This architecture uses a combination of secure overlay tunneling, routing via consistent hashing, and filtering. A collaborative DDoS defense system proposed in [32] consists of routers which act as gateways. They detect DDoS attacks, identify and drop attack packets. Gateways are installed and communicate only within the source and the victim domains. In [31] the distributed defense system for protecting web applications from DDoS attacks is described. This system is deployed in both victim and attacker source end. In DefCOM (Defensive Cooperative Overlay Mesh) [23], a peer-to-peer network of cooperative defense nodes is used. When an attack occurs, nodes close to the victim

detect this and alert the other nodes. Core nodes and those in vicinity of attack sources suppress the attack traffic through coordinated rate limiting. DefCOM nodes are classified into three categories: Alert generator nodes detect the attack and deliver an alarm to the rest of the peer network; Rate limiter nodes limit traffic; and Classifier nodes differentiate between legitimate and attack packets. In [2] two perimeter-based defense mechanisms are suggested. These mechanisms rely completely on the edge routers to cooperatively identify the flooding sources and establish rate-limit filters to block the attack traffic.

In our approach attack and defense systems are represented as antagonistic teams of agents. The purpose of attack team consists in defining vulnerabilities and implementation of security threat directed on availability of network resources by executing DDoS attacks. The purpose of defense team is protection of computer network and own components from DDoS attacks. We have a goal to model and simulate different defense mechanisms. Agents of antagonistic teams compete to reach opposite intentions. Agents of the same team cooperate to achieve common intention (to fulfill attack on computer network or to defense the network).

3 Modeling and Simulation of Attack Agents' Team

The main idea of DDoS attack is "denial of service" of some network resource by joint operations of many components acting on attack side. Thus, the original task of violation of a resource availability is divided into a set of simple subtasks of "denial of service" that are ordered to particular specialized components. On the upper level the joint goal remains the same for all components. On the lower level the sub-goals are formed. Their achievement is needed to solve the joint goal. The components are interacting to coordinate local solutions. This is needed to achieve the required quality of joint "denial of service" solution.

The components of DDoS attack system are software entities having at least the following properties: autonomy; joint goal and the list of actions for this goal achievement; initial knowledge (about itself, interacting entities and environment) given by the developer; soft knowledge or tough algorithms that permit to process input data; communication and interaction mechanisms for joint goal achievement. These properties allow considering each component of the attack system as an intelligent agent and a set of them as an agents' team.

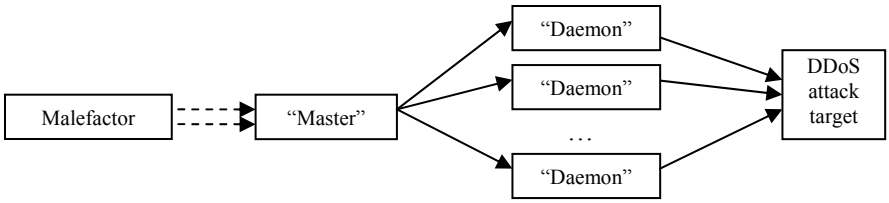


Fig.1. The two-level structure of attack team

Analysis of current DDoS methods allows choosing two main types of attack components: (1) coordinators of other components and (2) direct DoS executors.

Hence we select two classes of agents: master and daemon. The master is managing the sub-team of daemons. If for representation of an attack team we do not use several hierarchical levels, the master is only one in the team. Its function is coordination of daemons' actions. Thus master is playing a coordinator role. Daemons are doing actual attack actions. They can exchange information for joint goals achievement. Daemons play an executor role. So the attack team is a two-level system (figure 1). Masters act on the higher level directly fulfilling the malefactor's tasks. They make decisions: when to start the attack, what target to attack, what is the attack intensity. Masters coordinate daemons' actions by sending commands. Daemons act on lower level. After receiving the messages from masters, they start or finish sending the attack packets or change the attack intensity.

On the initial stage, the master and daemons are deployed on the available (compromised) hosts in the Internet. The malefactor sets the joint team goal – to perform DDoS attack against some of network resources. The master generates the parameters of attack and sends them to available daemons. Further daemons act. Their local goal is to execute commands. So they are sending attack packets to the given host. It is believed that the team goal is achieved if the given resource is completely or partially inaccessible (it deepens on the malefactor goal). Daemons periodically send to the master the messages that they are able to work. While receiving these messages the master controls the attack rate. If there is no message from one of daemons then the master makes decision to change the attack parameters, for example, the intensity of attack. Master can implement it by sending a command to change intensity. The malefactor can stop the attack. In this case he/she sends to the master the command "finish the attack". Then the master sends the corresponding commands to daemons. After receiving these commands they stop the attack.

The developed ontology of attack team comprises a hierarchy of notions specifying the activities of attack team on different layers of detail. In this ontology, the hierarchy of nodes splits into two subsets according to the macro- and micro-layers of the domain specifications. All nodes of the ontology are divided into intermediate and terminal. The nodes specifying a set of methods for generating DDoS attacks and main network notions make up a top level of the ontology. At lower levels, different classes of DoS-attacks and notions needed to implement the attack are detailed. A low-level fragment of attack ontology representing the classes of agents and their properties is depicted in figure 2. The low-level fragments of ontologies have been developed using MASDK toolkit [9].

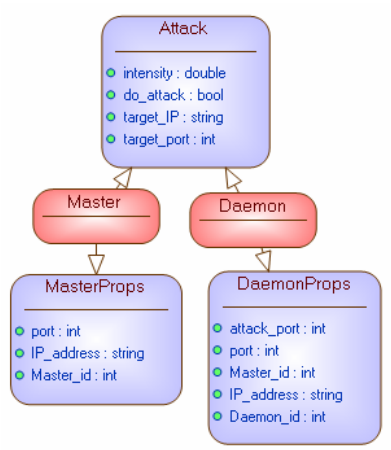


Fig.2. The fragment of attack ontology (the screenshot of MASDK ontology editor)

In the phase of deploying the agents, they get the following properties: (1) Master properties (*MasterProps*): the port for interaction, the IP address of agent's host; (2) Daemon properties (*DaemonProps*): the port for attack execution, the port for interaction, the IP

address of agent's host. The agents are identified by the attributes *agent_class_name_id*. The identifiers are used for interacting between the agents. The master forms the attack parameters that are defined by the *Attack* notion. The attack parameters are: intensity, start of attack (yes/no), the IP address of attacked host and its port. The daemon stores and uses these parameters.

Agent "Daemon": "Executor" Role. The daemon tries to receive and execute the master's commands. Depending on situation its local goal may be: receive a command from master; execute a command (start the DoS attack on the given host; finish the attack; change the attack intensity); send to the "master" the message about ability to work.

The agent can interact with the master in a coordinator role. In figure 3 it is represented what protocol can be used for interaction of agents with different roles.

The master plays a coordinator role (this fact is marked in figure by rhomb on the cross of lines directed from *Master* and *Coordinator*). The master initiates interaction (in figure this is marked by triangle on the cross of lines directed from *Master* and *AttackProtocol*).

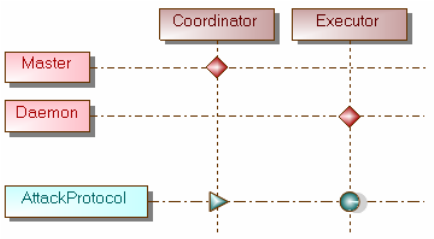


Fig.3. The meta-model of attack team (the screenshot of MASDK meta-model editor)

In the developed prototype the message from the masters to daemon has the following format:

Start the attack: yes/no	IP address of attack target	Port of attack target	Intensity of attack (in packets per second)

Agent in an executor role receives the message while listening to the given TCP port. If the "start the attack" field has the value "yes" the agent starts the DoS attack on the given host with given intensity. The message with label "no" serves for stopping the attack. The resources of executor role are a part of host resources, including processor time, memory domain and network resources used by given ports. When losing the control on one of these resources the agent stops to function.

It is necessary to assign a set of parameters that affects on the agent behavior with given role and also a set of variables that describes this process. The execution of DoS attack "UDP flood" for agent in an executor role consists in sending UDP-packets on the given address with given intensity (rate). The main variables that describe the agents' activity are the IP address and port of attack target and the intensity of packet sending. The intensity can be changed for lowering channel bandwidth without its denial of service. Port numbers are set when the agent starts working.

Agent "Master": "Coordinator" Role. The master coordinates the activities of agents playing a role executor. Depending on situation the local goals of master can be: receiving the commands from the malefactor; forming the messages for agents in the executor role; processing the messages about the state of agents in the daemon role; sending the messages for agent in the executor role.

The agent can interact with the agent of daemon class in an executor role (figure 3). The master acts as an initiator of interaction. The daemon needs the following data

from malefactor: the moments of time to start and stop the attack; the IP address and port of attacked host; the intensity of attack. These parameters can be given by malefactor due to user interface. Further the master forms the message and sends it to executors. He knows beforehand the addresses and listened ports for all agents playing an executor role.

4 Modeling and Simulation of Defense Agents' Team

The analysis of defense systems against DDoS attacks allow to discover the following their important features: (1) Defense systems are built on the basis of various components (sensors, filters, load balancers, queues, etc.) that fulfill different subtasks but serve to joint defense task; (2) The component set and functionality of defense systems depend on the place of the system deployment (external network, routers, gates, firewalls, internal network, servers, etc.); (3) Defense systems have several processing levels (initial processing of traffic, generating statistical information, detection, filtering, load balancing, tracing, etc.) on which the particular sub-tasks of complex defense task are solved.

The common approach to defense against DDoS attacks consists in the following. Sensors collect information about the normal network traffic for building a model of normal traffic. Then a special analyzer compares in real-time the current traffic with the model of normal traffic. The defense system tries to detect anomalies, trace back their sources and generates the recommendations of how to cut off or to lower the malicious traffic. The system administrator chooses countermeasures that are implemented by protection components.

The agents of defense team have the common joint goal: to defense the given host or network from a DDoS attack. According to common approach, the following classes of defense agents can be set (figure 4): "Sensor" (agent of initial processing of information); "Detector" (agent of attack detection); "Filter" (agent of filtering); "Investigator" (agent of investigation and forensic). Each class of agents is represented in the figure only by one instance.

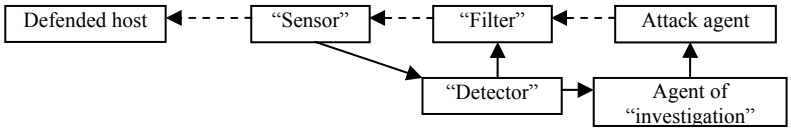


Fig.4. The generalized structure of defense team

In the initial stage the agents are deployed on the corresponding to their roles hosts: sensor – on the way of traffic passing to defended host; detector – on any host of defended subnet; filter – on the entrance to the defended subnet; investigator – on any available host beyond the defended subnet. The defense team includes a number of sensors. Sensor process information about network packets, collects statistic data, determines the size of overall traffic and the addresses of *n* hosts that make the greatest traffic. The detection agents (detectors) make decision if a danger of DDoS attack exists and what hosts are the attack sources. The filtering agents (filters) can

use different mechanisms of filtering the malicious packets. Investigators try to trace back the sources of DDoS attacks and neutralize them by defeating ("killing") corresponding attack agents. In addition, there can be used also the agents – managers interacting with a security administrator and configuring the defense system.

The developed ontology comprises a hierarchy of notions specifying activities of team of defense agents directed to protection from attacks on different layers of detail. The nodes determining the high levels of defense mechanisms (system, network, global) and main network notions make up the top level of the ontology. The high level fragment of DDoS defense mechanisms ontology is depicted in figure 5. At the bottom levels of the ontology these nodes are described by particular defense mechanisms [21, 33]. Different types of nodes corresponding to system level defense mechanisms can be used. For example, scanning tools check presence of DDoS-agents in the host file system, and also scan the ports frequently used by attackers. Mechanisms of client bottlenecks are directed on creating bottleneck processes on the zombie hosts used for DDoS-attacks to limit their attacking ability. Mechanisms of moving target defense consist in changing IP address to avoid being attacked.

The low level fragment of defense ontology is represented in figure 6. The goal of defense team is to protect the host *TargetHost* with given IP-address from DDoS attacks. The notion *TargetPort* contains the open ports of host *TargetHost*. The attribute *host_id* is the host identifier of the host. The attributes of notion *agent_name_(class)_Props* (including *InvestigationProps*, *FilterProps*, *DetectorProps* and *SensorProps*) for all agents are as follows: IP address of agents' host (*IP_address*) and the port for interaction (*port*). The attributes *agent_name_(class)_id* are the agent identifiers.

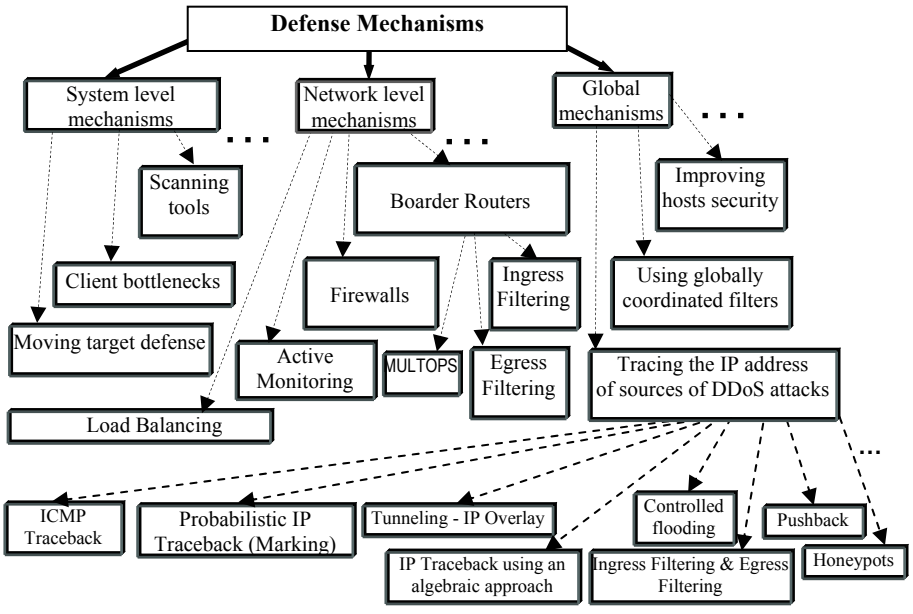


Fig.5. The high-level fragment of DDoS mechanisms ontology

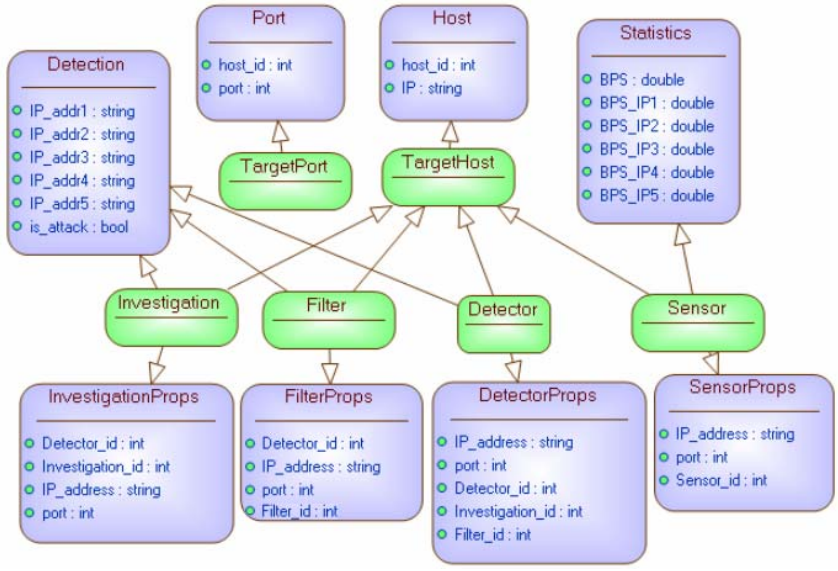


Fig.6. The low-level fragment of defense ontology (the screenshot of MASDK ontology editor)

Agent "Sensor": "Collector of Statistics" Role. Sensor can have the following sub-goals: initial processing of network packets; statistics collection by calculation of input traffic parameters from all external hosts together and separately; determining of the hosts that make the most intensive traffic; forming and sending the messages to detector about traffic statistics and chosen hosts. Figure 7 shows what protocol can be used for interaction. Sensor is an interaction initiator. Sensor forms the new portion of statistics and determines the five hosts with the greatest traffic every *k* seconds. If traffic from these hosts would be blocked later, the agent will determine next five hosts. Then the addresses of the hosts and amount of their traffic are sent to detector. The amount of overall traffic is also sent. The address and the port of receiver are known beforehand.

Agent "Detector": "Detector" role. The goal of the detector agent is to detect the attack. The local sub-goals may be: receiving and processing the messages from sensor; making decisions about detection of attack; forming and sending the messages to other agents. Playing this role the agent can interact with sensor in role "statistics collector", filter and investigator in corresponding roles. The agent initiates the interaction with the last two agents (see figure 7). Every *k* second a detector receives from sensor the statistics about traffic. If detector determines that BPS is more than 80% from maximum channel bandwidth, it determines the DDoS attack. The detector sends its decision and five addresses to filter and investigator. The addresses and ports of recipients are known beforehand.

Agent "Filter": "Filter" Role. The agent filters the traffic dropping malicious packets directed to defended host or net. The local sub-goal may be: receiving the messages from detector; filtering packets on the basis of data from detector. The agent can interact with detector. The initiator of interaction is detector (see figure 7). Every *k* seconds a filter receives from detector the data. If it is said in the message that

DDoS attack takes place, then filter begins to drop the packets from given IP addressee. Filter is a program that is deployed on the router (on the entrance to defended subnet).

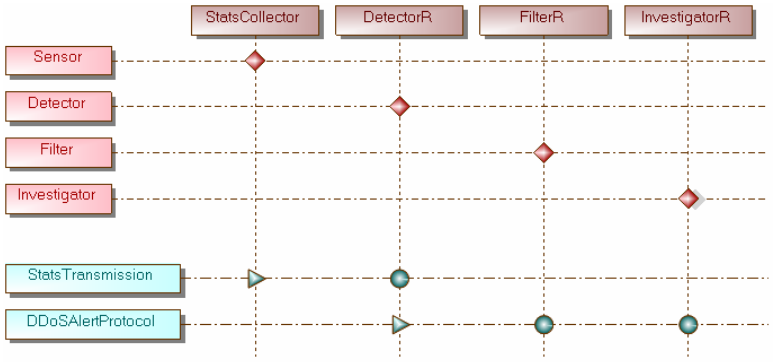


Fig.7. The meta-model of defense team (the screenshot of MASDK meta-model editor)

Agent of "Investigation": "Investigation" Role. The agent goal is to identify the attack agents and to defeat them. The investigator examines if the hosts with given addresses contain the attack agents and tries to "kill" them. Thus the local sub-goals may be: receiving and processing the messages from detector; search for suspicious hosts; defeating the attack agents on suspicious hosts. Playing this role the agent can interact with detector. Detector serves as interaction initiator (see figure 7).

5 Case-study: Example of Attack and Defense Simulation

To choose the network simulation tool the analysis of existing network simulators (including NS2, OMNeT++ [24], SSF Net, J-Sim and others) was made. The main requirements that were used to choose a network simulator are as follows: the detailed implementation of various protocols that are engaged in DDoS attacks (from the network layer and higher) in order to implement the main known DDoS attacks; the possibility to write and attach own modules for implementing the agent-based approach; the possibility to change the simulation parameters during simulation; the simulator implementation for Windows and Linux and requirement of cross-platform; advanced graphic user interface; free for use in scientific research.

In the paper we present the results of network simulation generated by using the OmnetPP INET Framework.

One of network fragments used for simulation is depicted in figure 8. The hosts designated as "cli[i]" are personal computers connected to the Internet. The connection speed is 100Mb/s. The hosts designated as "r-j" are routers. They are connected to each other with speed 512Mb/s. "Srv" is a (Web) server available to clients from the Internet. It has the services on TCP and UDP protocols. "Firewall" is a firewall deployed for secure access to the server "srv" from the Internet. The connection speed of "firewall" and "srv" is 100Mb/s.

On the initial phase of simulation the “Attack” and “Defense” teams are created. The “attack” team contains five agents of class “daemon” and one agent of class “master”. The “daemons” are deployed on the hosts “cli[1]”–“cli[5]”, “master” – on the host “cli[0] (figure 8).

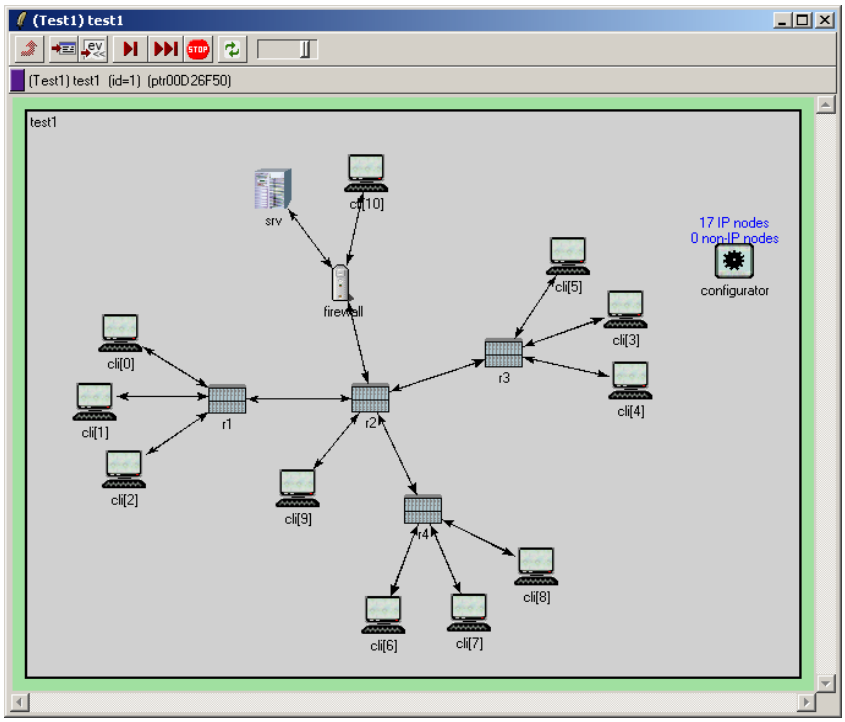
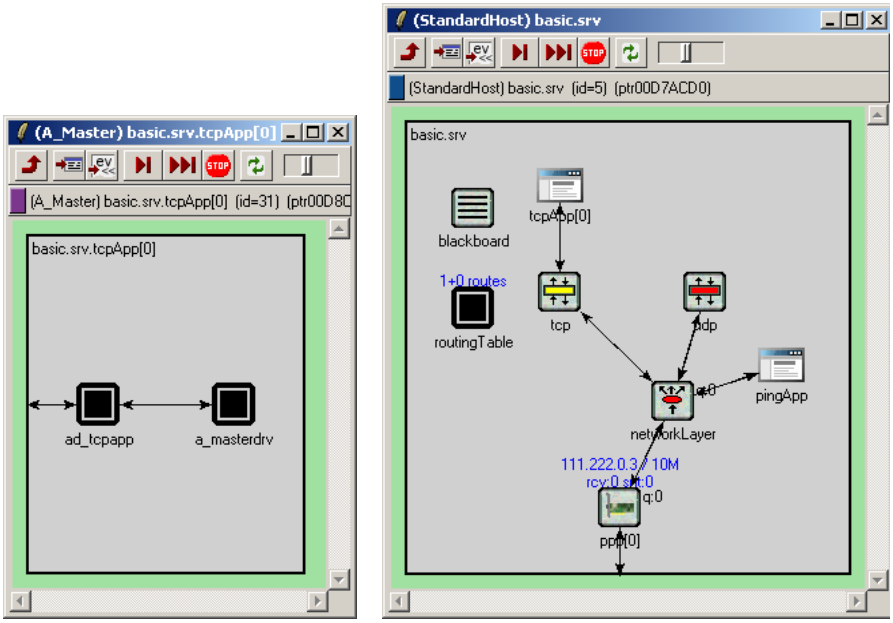


Fig.8. Structure of computer network used for simulation

Master contains the following modules (fig. 9a): ad_tcpapp – a TCP application controlled by the agent; a_masterdrv – a driver managing all components included in agent structure. Each “master” host contains the following modules (figure 9b): Ppp – is responsible for the data link layer; networkLayer – is responsible for network layer; pingApp – is responsible for applications that use ICMP; tcp – is the module serving TCP; udp – is the module serving UDP; tcpApp[0] – is agent “master”. As it is showed in figure 9b, the “master” host has IP address «111.222.0.3» and connection speed 10Mb (marked with blue font). The amounts of received (rcv:0) and sent (snt:0) packets are also depicted.

The defense team consists of two “sensors”, “detector”, “filter” and agent of “investigation”. The host under defense is “srv”. The first “sensor” is deployed on the host “r2”, the second – on the host “firewall”, “detector”– on the host “cli[10]”, “filter” – on the host “r2” and the agent of “investigation” – on the host “cli[9]”. The first “sensor” processes the traffic which is external for the defended sub-network (before the “r2” router, “filter” and “firewall”). The second “sensor” processes the internal traffic (after “filter” and “firewall”).



(a) (b)

Fig.9. The representation of agent "master" and the host where "master" is deployed

The defense agents are deployed similarly to the attack agents (figure 9). All defense agents use the TCP port #3000 for the interaction. All attack agents use the TCP port #2000 for interaction and "daemons" use UDP port #2001 for sending the attack packets. Hosts "cli[6]"–"cli[8]" are used for generating the background traffic between legitimate clients and server. The size of background traffic is not more then 20% of server channels bandwidth.

After creating the agents' teams in some moment of time, the malefactor sends to the "master" the command to attack the host with IP address "145.236.0.17" ("srv") on the port "17" with intensity "100 packets per second". Then the "master" sends the attack command to all "daemons". "Daemons" receive these messages. On the basis of received data, the corresponding parameters of "daemons" are changed (figure 10).

The main parameters that define the "daemons" attack are as follows: *local_port* – the port for sending the attack packets; *dest_port* – the port of the attacked host; *message_length* – the length of attack packet in bits; *message_freq* – the time interval between sending the packets; *dest_addresses* – the IP address of the attacked host.

If the "UDP flood" attack is set then "daemons" begin to send UDP packets to the server "srv". In figure 11 these packets are marked with yellow arrows. The red circle is a packet just sent by a "daemon" from host "cli[4]" to "srv". While performing attack, "daemons" are sending periodically to the "master" messages about their ability to work. Receiving the messages from "daemons", the "master" controls the progress of DDoS attack.

During functioning the defense agents' team, "sensors" are sending to the "detector" with given rate (for example, every 10 seconds) the statistic data about

traffic. In figure 11, the yellow arrow designates packets with the statistic data sent from the “sensor” located on the “firewall” to the “detector” on the host “cli[10]”.

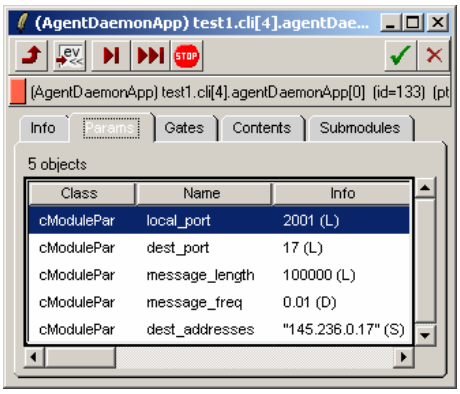


Fig.10. The “daemon” parameters after receiving the message from the “master”

After the beginning of attack, a sudden increase of malicious traffic (directed to “srv”) happens. The *BPS* parameter exceeds 80% from channel 100Mb bandwidth.

The “detector” receives the information about the network situation from “sensors”. This information contains *BPS* and the list of the five most active hosts. As *BPS* is exceeded, the “detector” makes decision about attack detection and sends the message about this fact to the “filter” and the agent of “investigation”.

The “filter” starts to drop the packets directed from “cli[0]”–“cli[5]” to “srv”. This is the reason why the size of internal subnet traffic (after “filter”) lowers.

After a short period of time, on the basis of data from internal “sensor”, the “detector” makes decision that there are normal traffic conditions (*BPS* < 80% from 100Mb). But on the basis of data from external “sensor” (deployed on the “r2” host), the “detector” “sees” that the attack is not finished.

The agent of “investigation” tries to discover and defeat the attack agents. At some moment of time, it succeeded in discovering the tracks of two attack agents deployed on the hosts “cli[1]” and “cli[3]”. Later the agent of “investigation” (cloning on the host “cli[3]” the components of defeating attack agents) succeeds to defeat (“kill”) the agent “daemon” located on the host “cli[3]”.

As a result, the malicious traffic directed to “srv” (before “filter”) essentially lowers. The agent “detector” changes the rules of filtering for the agent “filter”. At this moment the joint goal of defense team is considered as fulfilled.

After the fixed period of time the “master” does not receives the message about ability to work from “daemon” from the host “cli[3]”. Therefore to restore the needed intensity of attack, the “master” makes decision to increase the intensity of sending the malicious packets by other “daemons”. The “master” sends to the residuary “daemons” the attack command with enlarged intensity parameter. “Daemons” change the attack rate. This change causes the corresponding reaction of defense agents. The simulation continues further (till the given moment of time) representing the different “steps” of counteracting agents’ teams.

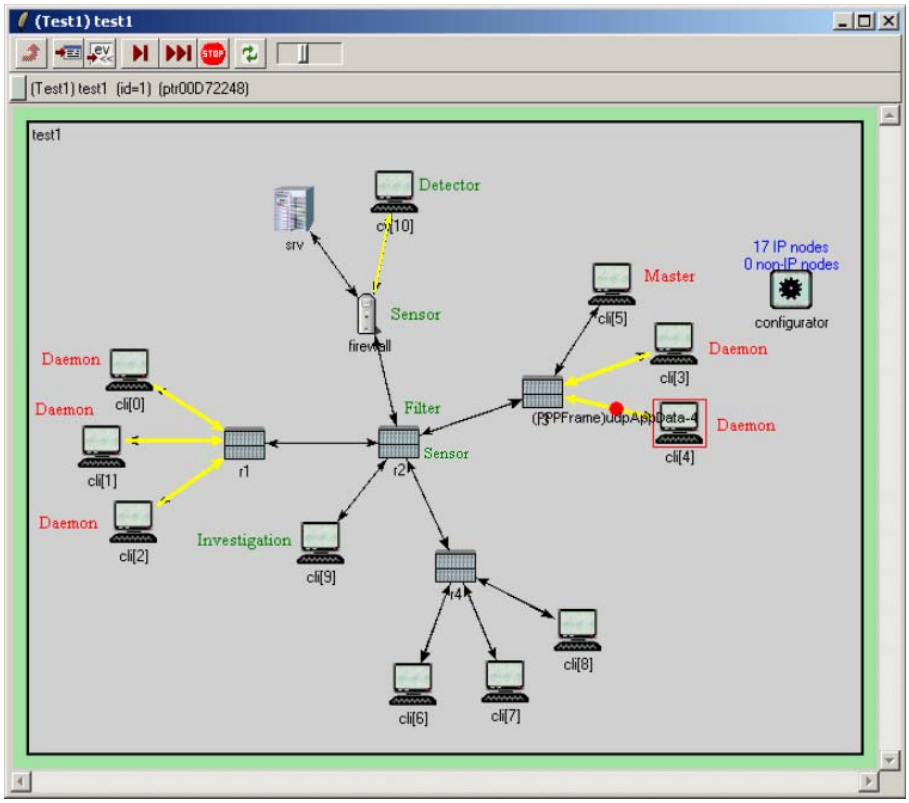


Fig.11. Representation of initial stages of the attack

6 Conclusion

In the paper we investigated basic ideas of the modeling and simulation of agents' competition in the Internet between the teams of software agents for network resources availability. The technology for creation of the agents' team was suggested and described. We developed the approach to be used for investigating different aspects of competition of teams in network environment. We presented the structure of a team of agents, specifications of hierarchies of agent plans, agent interaction-and-coordination mechanisms, and agent role-assignment mechanisms. Software prototype was developed using the OmnetPP INET Framework [24], Visual C++ and MASDK [9]. We imitated different classes of DDoS attacks and defense mechanisms. Experiments with the prototype have been conducted, including the investigation of attack scenarios against networks with different structures and security policies.

Our future theoretical work is directed on development of formal basis for agent-based modeling and simulation of antagonistic agents' teams on an example of agents' competition in the Internet. Our future practical goal is conducting experiments to both evaluate computer network security and analyze the efficiency

and effectiveness of security policy against different DDoS attacks. So the further development of our modeling and simulation framework and software tools will consist of developing more realistic environment, including improvement of capabilities of the attack and defense agents teams by expansion of the attack and defense mechanisms classes, and implementing more sophisticated attack and defense scenarios.

Acknowledgment

This research is being supported by grants 04-01-00167 of Russian Foundation of Basic Research and partly funded by the EC as part of the POSITIF project (contract IST-2002-002314).

References

1. 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).
2. Chen, S., Song, Q.: Perimeter-Based Defense against High Bandwidth DDoS Attacks. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 7 (2005).
3. Cohen, P.R., Levesque, H.J.: Teamwork. *Nous,* Vol. 25, No. 4 (1991).
4. Ellison, R.J., Fisher, D.A., Linger, R.C., Lipson, H.F., Longstaff, T., Mead, N.R.: *Survivable Network Systems: An Emerging Discipline. Technical Report*. November (1997).
5. Fan, X., Yen, J.: Modeling and Simulating Human Teamwork Behaviors Using Intelligent Agents. *Journal of Physics of Life Reviews*, Vol. 1, No. 3 (2004).
6. Garstka, J.: *Network Centric Warfare: An Overview of Emerging Theory*, PHALANX, December (2000).
7. Giampapa, J., Sycara, K.: Team-oriented agent coordination in the RETSINA multi-agent system. *Tech. report CMU-RI-TR-02-34,* Robotics Institute, Carnegie Mellon University (2002).
8. Gil, T.M., Poletto, M.: MULTOPS: a data-structure for bandwidth attack detection. *Proceedings of 10th Usenix Security Symposium*, August (2001).
9. Gorodetski, V., Karsayev, O., Kotenko, I., Khabalov, A.: Software Development Kit for Multi-agent Systems Design and Implementation. *Lecture Notes in Artificial Intelligence,* Vol.2296, Springer Verlag (2002)
10. Grosz, B., Kraus, S.: Collaborative plans for complex group actions. *Artificial Intelligence*, Vol.86 (1996).
11. Grosz, B., Kraus, S.: The evolution of SharedPlans. *Foundations and Theories of Rational Agencies,* A. Rao and M. Wooldridge (Eds.) (1999).
12. Jennings, N.: Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, No.75 (1995).
13. Kang, M.: Team-soar: A computational model for multilevel decision making. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 31, No. 6 (2001).
14. Keromytis, A.D., Misra, V., Rubenstein, D.: SOS: An architecture for mitigating DDoS attacks. *Journal on Selected Areas in Communications*, Vol. 21 (2003).

15. Kotenko, I., Stankevich, L.: The Control of Teams of Autonomous Objects in the Time-Constrained Environments. *Proceedings of the IEEE International Conference "Artificial Intelligence Systems"*. IEEE Computer Society (2002)
16. Kotenko, I.: Teamwork of Hackers-Agents: Modeling and Simulation of Coordinated Distributed Attacks on Computer Networks. *Lecture Notes in Artificial Intelligence*, Vol.2691 (2003).
17. 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).
18. 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).
19. Mahajan, R., Bellovin, S.M., Floyd, S.: Controlling High Bandwidth Aggregates in the Network. *Computer Communications Review*, Vol.32, No.3 (2002).
20. Martin, D., Cheyer, A., Moran, D.: The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence,* Vol.13, No.1-2 (1999).
21. Mirkovic, J., Martin, J., Reiher, P.: *A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms*, Technical report #020018. University of California, Los Angeles (2002).
22. Mirkovic, J., Dietrich, S., Dittrich, D., Reiher, P.: *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR (2004).
23. Mirkovic, J., Robinson, M., Reiher, P., Oikonomou, G.: Distributed Defense Against DDOS Attacks. *University of Delaware CIS Department Technical Report CIS-TR-2005-02* (2005).
24. OMNeT++ homepage. http://www.omnetpp.org
25. 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).
26. Papadopoulos, C., Lindell, R., Mehringer, I., Hussain, A., Govindan, R.: Cossack: Coordinated suppression of simultaneous attacks. *Proceedings of DISCEX III* (2003).
27. Peng, T., Christopher, L., Kotagiri, R.: Protection from Distributed Denial of Service Attack Using History-based IP Filtering. *IEEE International Conference on Communications* (2003).
28. Stone, P., Veloso, M.: Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence,* Vol. 110 (1999).
29. Tambe, M.: Towards flexible teamwork. *Journal of AI Research*, Vol. 7 (1997).
30. Tambe, M., Pynadath, D.V.: Towards Heterogeneous Agent Teams. *Lecture Notes in Artificial Intelligence*, Vol.2086 (2001).
31. 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).
32. 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).
33. Xiang, Y., Zhou, W., Chowdhury, M. : A Survey of Active and Passive Defence Mechanisms against DDoS Attacks. *Technical Report, TR C04/02*, School of Information Technology, Deakin University, Australia, March (2004).
34. Yen, J., Yin, J., Ioerger, T.R., Miller, M., Xu, D., Volz, R.: CAST: Collaborative agents for simulating teamworks. *Proceedings of IJCAI'2001* (2001).
35. Zachary, W.W., Mentec, J.L.: Modeling and simulating cooperation and teamwork. *Military, Government, and Aerospace Simulation,* M. J. Chinni (Ed.), Vol. 32 (2000).