Software testbed and experiments for exploring counteraction of attack and defense agents in the Internet

I.V.Kotenko, A.V.Ulanov[1]

1. Introduction

At present the formalization of processes that occur in Internet is an important direction of scientific research in computer network security domain. The goal is to provide the appropriate defense mechanisms against present and newly appeared threats.

In the given context this problem can be considered as the problem of formalization of organizational and technical counteraction between information defense and offense systems. The solution of this problem can be based on the investigative modeling and simulation of the mentioned counteraction processes using the family of various models (from analytical to scaled-down (emulational) and full-scale) (fig.1).

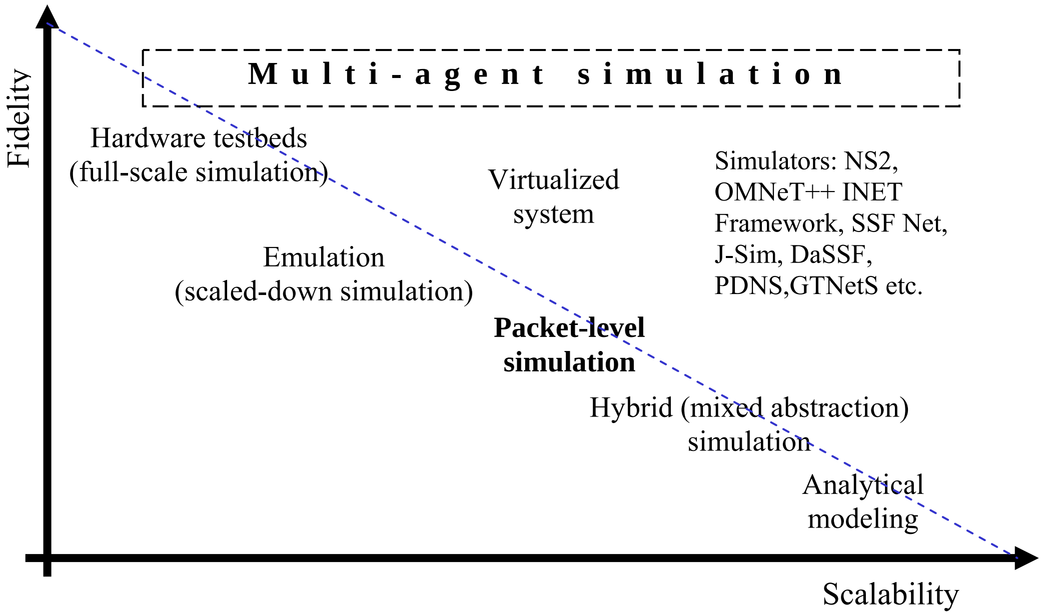

Multi-agent simulation
Fidelity
Scalability
Hardware testbeds (full-scale simulation)
Virtualized system
Simulators: NS2, OMNeT++ INET Framework, SSF Net, J-Sim, DaSSF, PDNS,GTNetS etc.
Emulation (scaled-down simulation)
Packet-level simulation
Hybrid (mixed abstraction) simulation
Analytical modeling

Fig.1. Family of models that are used for investigative modeling and simulation of computer network counteraction

In this paper we are developing an agent-oriented approach to the modeling and simulation of offense and defense systems' counteraction. This

[1] This research is being supported by grant of Russian Foundation of Basic Research (№ 04-01-00167), grant of the Department for Informational Technologies and Computation Systems of the Russian Academy of Sciences (contract №3.2/03) and partly funded by the EC as part of the POSITIF project (contract IST-2002-002314).

counteraction is represented as an antagonistic interaction between teams of software agents. The approach is stated in [1-3].

The main accent in the paper is given to two main aspects:

(1) the presentation of developed software environment (testbed) for multi-agent modeling and simulation of mentioned counteraction based on the principles of packet-level simulation (fig.1) and

(2) the description of the experiments on imitation of distributed denial of service attacks (DDoS) (targeted to the violation of information resources availability) and defense mechanisms realizing attack detection, prevention and pro-active reaction.

2. The approach to modeling and simulation

The multi-agent approach for modeling and simulation of defense processes in the Internet supposes that the cybernetic counteraction is represented as the interaction of various teams of software agents [1, 2]. At least two agent teams are distinguished: the team of agents-malefactors and the defense team. They act upon computer network and each other. The agents from different teams compete to achieve contrary intentions. The agents of the same team collaborate to achieve a joint intention.

The global goal for each team is achieved by the joint efforts of many components. The components of each team have the following features: autonomy; the presence of knowledge about itself, interacting entities and the environment; the presence of knowledge or the hard-coded algorithm allowing to get and process the external data from the environment; the presence of the goal and the list of actions to achieve this goal; the fulfillment of communications for achieving the common goal.

There are a number of approaches for organizing agent teamwork. The basic approaches are as follows: joint intensions theory [4], shared plans theory [5] and combined approach [6].

In the joint intentions theory the agent team has joint commitments and intentions. Agents have individual commitments that are their permanent goals. The individual intention of each agent is to achieve the goal.

The team plan is the basis of shared plan theory. This plan assigns the joint fulfillment of some set of team actions. The agent team has to reach the agreement on team action fulfillment.

The combined theory unites two first approaches.

A lot of teamwork approaches are implemented in differnt multi-agent systems. GRATE* [7] is the implementation of joint responsibilities teamwork. The following notions are at the heart of OAA [8] framework: “blackboard” for agent communications and “facilitator” that manages it. The main idea of CAST [9] is to use the shared mental model of agents for pro-active information exchange to achieve an effective teamwork. It is supposed in RETSINA-MAS [10] that every agent has the personal copy of partial plan. This copy lets them to estimate their abilities and to choose the corresponding roles. In “Robocup Soccer” [11] agents have the joint rules and knowledge and also the individual world models. These features manage their cooperative behavior.

COGNET/BATON [12] is the system for modeling the teamwork of people using intelligent agents.

The proposed approach for teamwork is based on the joint use of the elements of the joint intentions theory, shared plans theory and combined approach. It also takes into account the experience on realization of multi-agent systems.

The structure of agent team is described in terms of hierarchy of group and individual roles [1]. The leaves of the hierarchy correspond to the roles of particular agents and the intermediate nodes – to the group roles. The mechanisms of agent interaction and coordination are based on the following three groups of procedures: (1) action coordination; (2) monitoring and recovering of agent functionality; (3) communication selectivity (for the choice of the most "useful" communication acts).

The specification of action plan hierarchy is made for each of the roles. For every plan the following elements are described: the initial conditions, when the plan is proposed for execution; the conditions under which the plan is ended; the actions that are executed on the team level as a part of shared plan. The group plan has joint actions.

3. DDoS attacks and defense mechanisms

The proposed approach to the multi-agent modeling and simulation of computer network counteraction was proved on the basis of DDoS attack and defense mechanisms against them.

The main idea of DDoS attack is that the global goal – "the denial of service" of some resources – is accomplished by the joint operations of many components acting on the attack side. Thus the original task on DDoS is divided into simple subtasks that are ordered to particular specialized components. On the top level the goal remains the same for all components. On the lower level the local goals are formed. Their achievement is needed to solve the joint goal. The components are interacting with each other to coordinate the local solutions. This is needed to achieve the required quality of joint goal solution.

There are several kinds of DDoS attacks. They can be divided into two categories: exhaustion of network resources and exhaustion of host resources. The attacks are fulfilled by sending to the victim the large amount of packets (for example, UDP flood, ICMP flood, and also Smurf, Fraggle – via intermediate hosts), too long packets (Ping of Death), incorrect packets (Land), the large amount of laborious requests (TCP SYN), etc.

Building of effective defense system against DDoS is a very complex task. The usual measure to defense the subnet (not only from the DDoS attacks) is to apply the filtering rules for the packets from reserved IP addresses, protocols and ports (for example, for the incoming packets with the addresses from the internal pool, for the outgoing packets with the addresses not from the internal pool, for the packets to/from the unused ports, for the packets using unused protocols, etc.). Furthermore, the limitation on traffic for every protocol and for input/output streams can be applied.

Knowing this measures the malefactor can use such parameters of DDoS attack that it will be impossible to distinguish the attack from, e.g., the users requests caused by an increased interest to the given server. This complicates defense mechanisms.

The common approach to defense against DDoS is as follows. The information about the normal traffic for this network is collected by sensors. Then the component-analyzer compares in real-time the current traffic with the model of normal traffic. The system tries to trace back the source of abnormalities (with the help of “traceback” mechanisms) and shows the recommendations of how to sever or to lower them. The system applies the countermeasures the system administrator (or the system user) chooses.

It can be distinguished two main tasks of defense systems: attack detection and attack counteraction.

The mechanisms of attack detection can be classified by the place of deployment and by the method of detection. The components of detection can be deployed in the attacked, the source or the intermediate sub-networks. The attack detection occurs due to the comparison of the current and model traffic. The model of normal network traffic is created using the available traffic data: either evidently, or after processing by some method. As a rule, this model is based on the load [13, 14, 15, 16, 17], on the signature [18, 19, 20], on the statistics [21, 22, 23, 24, 25, 26, 16, 17, 27], with the use both standard statistical methods and other methods (e.g., due to hierarchical system of various classifiers which can learn [30]).

The mechanisms of DDoS attack counteraction can be classified as detection mechanisms taking into account the place of deployment and the defense method used. The place of deployment is determined by the defense target. This can be the attacked, the source or the intermediate sub-networks. Besides own protection, the system of effective counteraction influences also positively on the remaining network as a whole, e.g., by blocking the attack packets within itself. The defense methods may be as follows: packet filtering (it is used in the most cases), flow filtering [26], changing the amount of resources [32, 33, 34, 27], authentication [13, 31, 35], etc.

Additionally three variants of applying the packet filtering can be distinguished. The first (traditional) variant is a standard filtering preformed on one host. The second variant is with “pushback” [14, 26, 15, 16, 17] when the filter is applied on every iteration nearer to the attack source. The third – is with “traceback” [36, 37, 38, 22, 39, 23, 24] when the source of attack is traced and the filter is applied on the nearest host (on the router).

4. Attack agent team

Attack agents are divided, at least, into two classes. They are “daemons” that realize the attack directly and “master” that coordinates the actions of other system components.

On the preliminary stage the master and daemons are deployed on available (compromised) hosts in the Internet. The important parameters on this stage are agents’ amount and the degree of their distribution. Then the attack team is established: daemons send to master the messages saying they are alive and ready to work. Master stores the information about team members and their state.

The malefactor sets the common goal of the team – to perform DDoS attack with some parameters. Master receives attack parameters. Its goal is to distribute these parameters among all available daemons. Then daemons act. Their local goal is to execute the master command. To fulfill attack they send the attack packets to the given host with the intensity (attack rate) appointed by master. After this it is believed that the goal on this stage of attack is reached.

Master asks daemons periodically to find out that they are alive and ready to work. Receiving the messages from daemons the master manages the given rate of attack. If there is no any message from one of the daemons the master makes the decision to change the attack parameters. For example, it can send to some or all daemons the commands to change the attack rate.

Daemons can execute the attack in various modes. This feature affects on the potentialities of defense team on attack detection, blocking, traceback and attack agents defeating. Daemons can send the attack packets with various rate, spoof source IP address and do it with various intensity.

Malefactor can stop the attack by sending to master the command “stop the attack”. Then master distributes this command among all daemons. When they receive this command they stop the attack.

5. Defense agent team

Corresponding to the general approach there are distinguished the following defense agent classes [3]: initial data processor (“sensor”); attack detection agent (“detector”); filtering agent (“filter”); investigation agent (“investigator”).

Let us describe the main functionality of these agents in one of the experiments described in the paper. In other experiments their functionality can be extended, and additional classes of agents can be deployed.

In the initial moment of time the defense agents are deployed on hosts according to their roles:

sensor is deployed on the way of traffic to defended host;

detector – on any host in defended subnet;

filter – on the entrance to defended subnet;

investigator – on any available host beyond the subnet.

The joint goal of defense team is to protect against DDoS attack. Detector watches on its accomplishing.

Sensor processes the information about network packets and collects statistic data on traffic for defended host. Sensor determines the size of overall traffic (*BPS – bit per seconds*) and the addresses of *n* hosts that make the greatest traffic (in developed prototype – all hosts). Its local goal is to give these parameters to detector every *k* seconds.

The local goal of detector is to make the decision that the attack happens. In experiments described in the paper the following method is realized. If detector determines that BPS is more than given rate (that is determined on the basis of amount of typical traffic for this subnet) than it decides that there is the DDoS attack. It sends its decision and the addresses of *n* hosts that make the greatest traffic to filter and investigator.

The local goal of filter is to filter the traffic on the basis of data from detector. If it was determined that the network is under attack, then filter begins to block the packets from the given hosts.

The goal of investigator is to identify and defeat attack agents. When investigator receives the message from detector it examines the given addresses on the presence of attack agents and tries to defeat identified agents. To simplify the model the admission is made that the defeating rate is 30%.

When detector determines (using data from sensors) that the attack is stopped, it believes that the joint goal of agent team is achieved on the given time interval.

6. Simulation environment

To choose the simulation tool the comprehensive analysis of the following software simulators was made: NS2 [40], OMNeT++ INET Framework [41], SSF Net [42], J-Sim [43] and some others. We used the following main requirements to the simulation environment: the detailed implementation of the protocols (from the network layer and higher) that are used in DDoS attacks (to simulate the main classes of DDoS attacks); the availability of writing and plugging in the new modules to implement the agent approach; free for use in research and educational purposes; advanced graphical user interface, etc. We discovered that the OMNET++ INET Framework satisfies to these requirements best of all.

OMNET++ is a discrete event simulator [41]. The events occur inside simple modules. The exchange of messages between modules happens due to channels (modules are connected with them by the gates) or directly by gates.

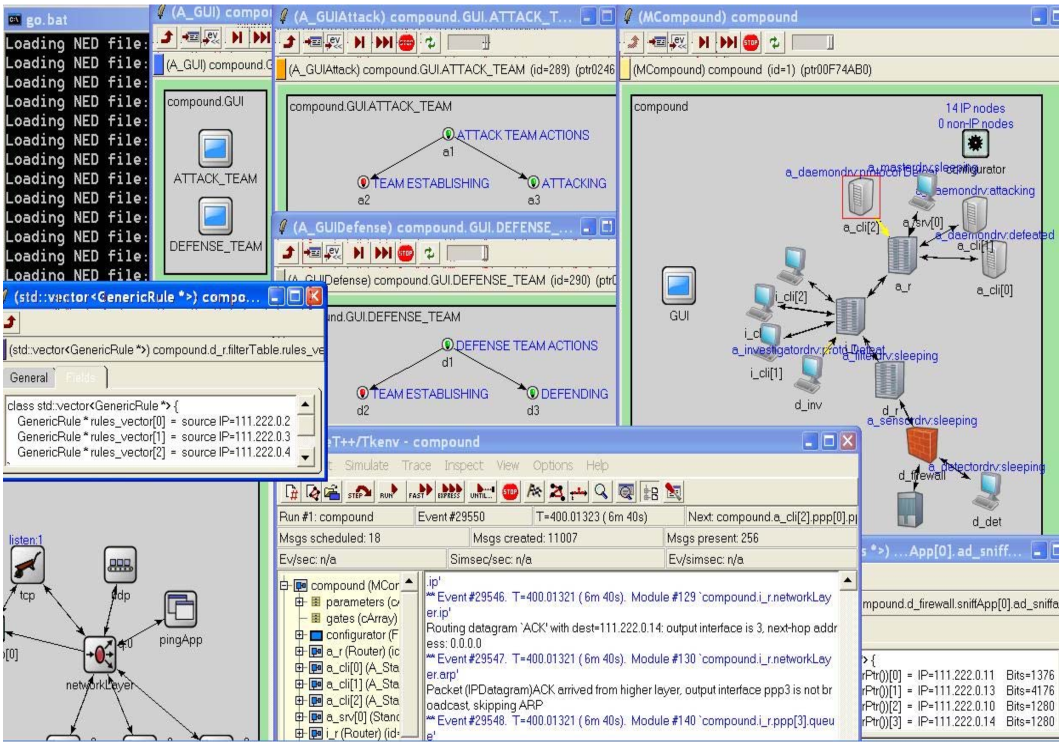
We are developing now the environment for multi-agent simulation of DDoS defense and attack mechanisms on the basis of OMNeT++ INET Framework. We have modified the existing OMNeT++ INET Framework. For example, the following new modules have been created: the filtering table for network layer (for defense actions modeling); the “sniffer” that allows to scan all traffic for the given host (to collect the statistics for simulation the defense side actions and also for attack actions simulation). The modules that provide “sockets” were changed to accurately simulate the attack mechanisms. The agent kernels were made as co-routines, as it is convenient for implementing the interaction protocols (on which the agent teamwork is based). The other modules were made as the handlers of events from the kernel and external environment.

The example of user interface of the simulation environment is represented in fig.2.

At the basic window of visualization (fig.2, at upper right), a simulated computer network is displayed. *The* network represents a set of the hosts connected by data channels. Hosts can fulfill different functionality depending on their parameters or a set of internal modules. Internal modules provide the corresponding protocols and applications at various levels of the OSI model. Hosts are connected by channels which parameters can be changed. Applications (including agents) are deployed on hosts by connecting to corresponding protocol modules.

The window for simulation management (at the bottom of fig.2, in the middle) allows looking through and changing simulation parameters. There are corresponding state windows that represent the current state of agent teams (at the top of fig.2, in the middle). There are available several information windows that depict the functioning (or statistics data) of particular hosts, protocols and agents. For example, the window of one of the hosts is represented in fig.2.

Each network for simulation consists of three sub-networks: (1) the subnet of defense where the defense team consisting from K hosts (including the defended hosts) is deployed; (2) the intermediate subnet where N hosts with generic clients are deployed; (3) the subnet of attack where the attack team is deployed, including M hosts with daemons and one host with master. The sizes of subnets may be set by the corresponding simulation parameters.



```
Loading NED file:
Loading NED file:
Loading NED file:
Loading NED file:
Loading NED file:
Loading NED file:
Loading NED file:
Loading NED file:
Loading NED file:
Loading NED file:
Loading NED file:
Loading NED file:
Loading NED file:
```

```
class std::vector<GenericRule *> {
  GenericRule * rules_vector[0] = source IP=111.222.0.2
  GenericRule * rules_vector[1] = source IP=111.222.0.3
  GenericRule * rules_vector[2] = source IP=111.222.0.4
```

```
Run #1: compound          Event #29550          T=400.01323 ( 6m 40s)          Next: compound.a_cli[2].ppp[0].p
Msgs scheduled: 18             Msgs created: 11007              Msgs present: 256
Ev/sec: n/a               Simsec/sec: n/a                       Ev/simsec: n/a
```

```
.ip'
** Event #29546. T=400.01321 ( 6m 40s). Module #129 `compound.i_r.networkLay
er.ip'
Routing datagram `ACK' with dest=111.222.0.14: output interface is 3, next-hop addr
ess: 0.0.0.0
** Event #29547. T=400.01321 ( 6m 40s). Module #130 `compound.i_r.networkLay
er.arp'
Packet (IPDatagram)ACK arrived from higher layer, output interface ppp3 is not br
oadcast, skipping ARP
** Event #29548. T=400.01321 ( 6m 40s). Module #140 `compound.i_r.ppp[3].queu
e'
```

```
> {
rPtr())[0] = IP=111.222.0.11   Bits=1376
rPtr())[1] = IP=111.222.0.13   Bits=4176
rPtr())[2] = IP=111.222.0.10   Bits=1280
rPtr())[3] = IP=111.222.0.14   Bits=1280
```

Fig.2. Example of user interface for simulation

8. Experiments

There were made several experiments using the models of DDoS defense and attack processes.

Let us examine one of simple simulation scenarios to demonstrate possibilities of the software environment developed. The network for this simulation scenario is represented in fig.2 (at the upper right). The routers in this network are connected with each other by fiberglass channels with bandwidth 512 Mbit. The other hosts are connected by 10 Mbit Ethernet channels.

Some time after the start of simulation, clients begin to send the requests to server and it replies. That is the way generic (normal) network traffic is generated.

The formation of defense team begins some time after the start of simulation. The defense agents (investigator, sensor and filter) connect to detector. They send to detector the messages saying that they are alive and ready to work. Detector stores this information to its knowledge base. The formation of attack team occurs in the same way.

The defense team actions begin after this team formation. Sensor starts to collect the traffic statistics (the amount of transmitted bytes) for every IP-address. Detector requests data from sensor every S seconds (e.g., 60 sec). It gets statistics and detects if there is an attack. Then it connects to filter and investigator and sends them the IP-addresses of suspicious hosts.

When attack actions begin, master requests every daemon if it is alive and ready to work. When all daemons were examined, it occurs that they all are workable. Master calculates the rate of attack for every daemon. Then master sends the corresponding attack command to every daemon. Daemons start the attack by sending, e.g., the UDP packets to the victim server with the given rate.

Sensors send to detector the list of IP addresses and the amount of bits transmitted for the given time interval. Detector determines which hosts (IP addresses) transmit the traffic that exceeds the maximum allowable size. Detector sends these addresses to filter to apply filtering rules and to investigator to trace and defeat the attack agents. After applying the filtering rules by filter the traffic to the server was lowered. And agent-investigator tries to defeat attack agents. It succeeds to defeat two of them. The remaining daemon continues the attack. Master redistributed the attack load for it. But the attack packets do not reach the goal and are filtered at the entrance of the defended network.

The dependence between traffic volume transmitted to the server subnet and time is represented in fig.3.

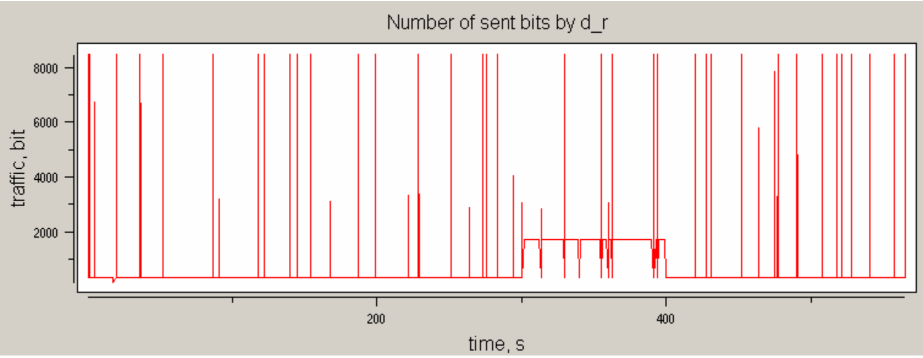


Fig.3. Dependence between the amount traffic and time

In time interval 0–300 seconds the main traffic was generated by the client requests to the server. This process is depicted by the vertical straights with low density. When the attack begins (the label of 300 seconds) the high-density traffic appears – the plateau between 300 and 400 seconds. But approximately at 400 second the filtering rules were applied and the attack packets begun to being dropped at the entrance to the server subnet. After that the normal state returns.

7. Conclusion

In this paper we described the multi-agent environment for modeling and simulation of counteraction between the teams of malefactor and defense agents in the Internet. The environment developed is written using C++ and OMNeT++. The various classes of attacks and defense mechanisms were implemented. A set of experiments was carried out on an example of modeling and simulation of attacks "Distributed Denial of Service". The experiments showed the effectiveness of the proposed approach and that it can be successfully used for modeling and simulation of prospective defense mechanisms and for security level analysis on the stage of network design.

Future work is connected with the further development of proposed counteraction models, including design and implementation of formal models of antagonistic interaction between the teams of defense and attack agents; implementation of greater amount of particular defense and attack mechanisms; evaluating effectiveness of implemented defense mechanisms; providing the recommendations on building of prospective defense systems against DDoS; further development of the simulation environment; investigation and improvement of agent teamwork mechanisms; developing the mechanisms of agent teams adaptation and self-learning.

References

1. I.V.Kotenko. Multiagent models of counteracting malefactors and security systems in Internet. Third All-Russian Conference "Mathematics and security of information technologies". Moscow, Moscow State University, 2004 (in Russian).
2. I.Kotenko. Agent-Based Modeling and Simulation of Cyber-Warfare between Malefactors and Security Agents in Internet. 19th European Simulation Multiconference "Simulation in wider Europe". ESM'05. 2005.
3. I.Kotenko, A.Ulanov. Multiagent modeling and simulation of agents' competition for network resources availability. Second International Workshop on Safety and Security in Multiagent Systems. SASEMAS'05. 2005.
4. P.Cohen, H.J. Levesque. Teamwork. Nous, No. 35, 1991.
5. B.Grosz, S.Kraus. Collaborative Plans for Complex Group Actions. Artificial Intelligence, Vol.86, 1996.
6. M.Tambe. Towards flexible teamwork. Journal of AI Research, Vol.7, 1997.
7. N.R.Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. Artificial Intelligence, Vol.75, No.2, 1995.
8. D.Martin, A.Cheyer, D.Moran. The open agent architecture: A framework for building distributed software systems. Applied Artificial Intelligence, Vol.13, No.1-2, 1999.

9. J.Yen, X.Fan, S.Sun, R.Wang, C.Chen, K.Kamali, M.Miller, R.Volz. On Modeling and Simulating Agent Teamwork in CAST. Proceedings of the Second International Conference on Active Media Technology, 2003.
10. J.A.Giampapa, K.Sycara. Team-Oriented Agent Coordination in the RETSINA Multi-Agent System. Tech. report CMU-RI-TR-02-34, Robotics Institute, Carnegie Mellon University, December, 2002.
11. L.A.Stankevich. A cognitive agent for soccer game. Proceeding of First Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS'99). 1999.
12. X.Fan, J.Yen. Modeling and Simulating Human Teamwork Behaviors Using Intelligent Agents. Journal of Physics of Life Reviews, Vol.1, No.3, 2004.
13. C.Sangpachatanaruk, S.M.Khattab, T.Znati, R.Melhem, D.Mosse. Design and Analysis of a Replicated Elusive Server Scheme for Mitigating Denial of Service Attacks. Journal of Systems and Software, Vol.73(1), 2004.
14. A.Keromytis, V.Misra, D.Rubenstein. SOS: Secure Overlay Services. Proceedings of ACM SIGCOMM'02, Pittsburgh, PA, 2002.
15. T.Peng, C.Leckie, R.Kotagiri. Defending Against Distributed Denial of Service Attacks Using Selective Pushback. 9th IEEE International Conference on Telecommunications, Beijing, China, 2002.
16. J.Ioannidis, S.M.Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. Proceedings of Symposium of Network and Distributed Systems Security (NDSS), San Diego, California, 2002.
17. R.Manajan, S.M.Bellovin, S.Floyd, J.Ioannidis, V.Paxson, S.Shenker. Controlling High Bandwidth Aggregates in the Network. ICSI Technical Report, July 2001.
18. Peakflow Platform. Arbor Networks. http://www.arbornetworks.com
19. DDoS-Guard. Green Gate Labs. http://www.ddos-guard.com
20. Prolexic Solutions. Prolexic. http://www.prolexic.com
21. C.Jin, H.Wang, K.G.Shin. Hop-count filtering: An effective defense against spoofed DDoS traffic. Proceedings of the 10th ACM Conference on Computer and Communications Security, 2003.
22. K.T.Law, J.C.S.Lui, D.K.Y.Yau. You Can Run, But You Can't Hide: An Effective Methodology to Traceback DDoS Attackers. Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems. MASCOTS'02. 2002.
23. A.C.Snoeren, C.Partridge, L.A.Sanchez, C.E.Jones, F.Tchakountio, B.Schwartz, S.T.Kent, W.T.Strayer. Single-Packet IP Traceback. IEEE/ACM Transactions on Networking, Vol.10, No.6, 2002.
24. J.Li, M.Sung, J.Xu, L.Li. Large-scale IP traceback in high-speed Internet: Practical Techniques and theoretical foundation. Proceedings of the IEEE Symposium on Security and Privacy. S&P'04. 2004.
25. J.B.D.Cabrera, L.Lewis, X.Qin, W.Lee, R.K.Prasanth, B.Ravichandran, R.K.Mehra. Proactive detection of distributed denial of service attacks using mib traffic variables – a feasibility study. Proceedings of International Symposium on Integrated Network Management, 2001.

26. D.Xuan, R.Bettati, W.Zhao. A Gateway-Based Defense System for Distributed DoS Attacks in High Speed Networks. Proceedings of the 2nd IEEE SMC Information Assurance Workshop, West Point, NY, June, 2001.
27. J.Mirkovic, G.Prier, P.Reiher. Attacking DDoS at the Source. Proceedings of ICNP 2002, Paris, France, 2002.
28. J.Kang, Z.Zhang, J.Ju. Protect E-Commerce against DDoS Attacks with Improved D-WARD Detection System. Proceedings of 2005 IEEE International Conference on e-Technology, 2005.
29. Y.Xiang, W.Zhou. An Active Distributed Defense System to Protect Web Applications from DDOS Attacks. Proceedings of the Sixth International Conference on Information Integration and Web-based Applications Services, iiWAS'2004. Jakarta, Indonesia, 2004.
30. V.Gorodetsky, O.Karsaev, V.Samoilov, A.Ulanov. Asynchronous alert correlation in multi-agent intrusion detection systems. Lecture Notes in Computer Science, Vol.3685, 2005.
31. X.Wang, M.K.Reiter. Mitigating bandwidth-exhaustion attacks using congestion puzzles. Proceedings of the 11th ACM Conference on Computer and Communications Security, 2004.
32. D.Mankins, R.Krishnan, C.Boyd, J.Zao, M.Frentz. Mitigating Distributed Denial of Service Attacks with Dynamic Resource Pricing. Proceedings of the 17th Annual Computer Security Applications Conference. ACSAC’01. 2001.
33. Y.Bernet, J.Binder, S.Blake, M.Carlson, B.Carpenter, S.Keshav, E.Davies, B.Ohman, D.Verma, Z.Wang, W.Weiss. A Framework for Differentiated Services. IETF Internet Draft, 1999.
34. H.Wang, S.G.Shin. Transport-aware IP Routers: A Built-in Protection Mechanism to Counter DDoS Attacks. IEEE Transactions on Parallel and Distributed Systems, Vol.14, No.9, 2003.
35. H.Wang, A.Bose, M.El-Gendy, K.G.Shin. IP Easy-pass: Edge Resource Access Control. Proceedings of IEEE INFOCOM`04, Hong Kong, 2004.
36. B.W.Gemberling, C.L.Morrow, B.R.Greene. ISP Security – Real World Techniques. Presentation, NANOG, October 2001.
37. Y.A.Perrig, D.P.Song. A path identification mechanism to defend against DDoS attacks. Proceedings of the 2003 IEEE Symposium on Security and Privacy, 2003.
38. S.Bellovin, M.Leech, T.Taylor. ICMP Traceback Messages. Internet-Draft draft-ietf-itrace-01.txt, October 2001.
39. S.Savage, D.Wetherall, A.Karlin, T.Anderson. Practical network support for ip traceback. Proceedings of the 2000 ACMSIGCOMM Conference, Stockholm, Sweden, August 2000.
40. NS-2 homepage. http://www.isi.edu/nsnam/ns/
41. OMNeT++ homepage. http://www.omnetpp.org/
42. SSFNet homepage. http://www.ssfnet.org
43. J-Sim homepage. http://www.j-sim.org