Dynamic Access Control Using Virtual Multicore Firewalls

Vladimir Zaborovsky, Alexey Lukashin
Department of Telematics
Saint-Petersburg State Polytechnical University
Saint-Petersburg, Russia
vlad@neva.ru lukash@neva.ru

***Abstract*—The problems of Internet services security are becoming particularly important due to intricacy structure and dynamic nature of distributed environment, especially in a cloud and virtualized systems. The complexity of distributed platforms demands more functionality to be provided by security devices. Among these required functions is the ability to configure these devices online in accordance with the current state of the network environment through which users can gain an access to information services. The performance of security services is a major issue. This paper proposes a firewall-based solution for implementing access control using multiple cores in virtualized and pure hardware environments, and describes dynamic access control based on virtual connections management with the mechanism of traffic filtering in a transparent (also called "stealth") mode. In this mode, the firewall is not visible to other participants (components) of network interactions, and, thus, it allows implementing the access policy, but remains invulnerable for cyber crooks.**

Keywords-security; dynamic access control; firewall; virtualization; netgraph

I. INTRODUCTION

Information security solutions like firewalls are very sensitive to the level of performance. Modern information channels support huge bandwidth, 10Gbit/s is almost everywhere. In 10Gbit/s Ethernet networks firewall has to make a decision in less than 1ms for a packet. During this time it should check protocol validness and pass all filtering rules to different network layers – from the channel to applied protocols. The current work proposes to achieve this goal by using multicore capabilities of modern computing platforms. The traffic processing should be made in parallel. Concurrent programming is quite complicated, so it is necessary to provide some generic approach which allows implementing parallel data processing for different cases. The paper describes another major issue – virtualization. Today, businesses of various sizes widely use virtualization. Small companies use cloud providers like Amazon or Rackspace, medium and big businesses have its own computing infrastructure based on virtualization. The main problem is that virtual systems are hidden from hardware security devices, like hardware firewalls; thus, the necessary "virtual" communication is usually not controlled. It is very important for cloud systems to find a solution for this problem; especially, for private cloud solutions such as Eucalyptus [1], OpenStack [2], OpenNebula [3] and others. Security is a very actual problem in the cloud [5, 6]. Modern government departments build their infrastructure using cloud systems and, of course, these systems should control all information resources. So, another requirement for modern firewall is the ability to be virtual as well as high performance. Firewall virtualization gives another opportunity that allows scaling firewall resources depending of the current situation by changing number of cores or memory in runtime. Firewall performance scalability is very useful for cloud systems. The nature of cloud environments is very dynamic; the resources, which can be presented in the cloud, are extremely different. The cloud firewall should also be dynamic and flexible, by having the possibility to reconfigure itself in runtime according to the current cloud state. In this paper, we propose a solution with parallel traffic processing models and describe architecture of cloud environment secured by virtual firewalls inside hypervisors. Our firewalls manage network traffic in stealth mode; the firewall interfaces haven't any physical addresses and invisible for other network components. It increases security and allows installing these firewalls transparently to hypervisor or physical network.

The main contribution of this paper is a graph virtual connection control model, which is implemented by using Netgraph [4] network subsystem. We also present a prototype of such stealth firewall which works as a separate hardware solution and as a virtual machine in hypervisor and manages virtual traffic.

The paper is organized in six sections. Section 1 is an introduction; the second one describes virtual connections and traffic filtering as computation graph. Section 3 describes virtual connection processing models using Netgraph network subsystem. Section 4 contains description of experiments and measurements. Section 5 proposes architecture of secure cloud with stealth multicore firewalls and Section 6 is the conclusions.

II. APPROACHES FOR VIRTUAL CONNECTIONS CONTROL

Packet flow is described as a set of virtual connections between users and services [7]. Virtual connection (VC) is a logically ordered exchange of messages between the network nodes. Virtual connections are classified as technological virtual connections (TVC) and informational virtual connections (IVC). A technological virtual connection is described by network protocols, e.g., TCP session between user and database. Information virtual connection is described by applied protocols, e.g., HTTP session with a

 ISBN: 978-1-61208-204-2

web service. IVC might use multiple TVC, e.g., ftp session uses 2 TCP connections; one for data and another for control messages. And vice versa, TVC might belong to multiple IVCs, e.g., persistent connections in HTTP, as described in RFC 2616 client can reuse existing TCP connections for multiple requests, of course, resource URI might be also different.

For flexible access control and traffic management dividing TVCs into three groups was proposed:

1) Permitted important connections without additional control;
2) Prohibited connections;
3) Other connections which are not prohibited yet but need additional control.

The first group is the priority connections and the third group is background connections. All packets of virtual connections in the second group are dropped by firewall and not taken into account.

We propose the preemptive priority queuing system with two types of packets [8]. First type has priority over the second one. The packets arrive into the buffer according to the Poisson process. The service time has the exponential distribution. The buffer has a finite size m and it is shared by both types of packets. The preemptive priority in service is given to the packets of the first type. Considered system is supplied by the randomized push-out mechanism that helps precisely and accurate to manage packets of both types. If the buffer is full, a new coming packet of type 1 can push out a packet of type 2 with the probability from the buffer.

As it is shown in [8], it is possible to change the time which packets spend in the firewall buffer by choosing α parameter. That allows to limit access possibilities of background traffic and even to block a connection if it's classified as being prohibited during the data transmission. The proposed mechanism also allows controlling TVC throughput and increasing time for the access decision without interrupting the established connection.

Technical virtual connection exists in parallel to and independently from other virtual connections. Virtual connections do not share any resources. It allows parallel processing of virtual connections. The suggested approach to the network traffic filtering is based on the concept of virtual connection and allows extracting the connection context. The connection context can be described as a vector Yi, which contains a set of parameters, for example, source and destination addresses, port, connection status (for TCP protocol), etc. Virtual connection control is a computation of the indicator function F, which requires resources, such as computing processors and operating memory.

$$F(Y_i) = \{1,0,*\} \quad (1)$$

The indicator function F takes the following values: 1- if VC is allowed; 0- if VC is forbidden; *, if at the current moment it is impossible to clearly determine whether connection is prohibited or not, the decision is postponed and VC is temporarily allowed.

Calculation of the indicator function F can be decomposed into multiple computing processes; $\{F_i\}$, $i=\{1..n\}$, where n is a number of independent calculation processes, e.g., evaluation of virtual connection might consist of filtering rules check, protocol validness check, intensity check, content check, etc. In this case, the problem of VC control can be described by using the graph G(Q,X), which is called the VC control information graph. Q is a set of nodes; X is a set of edges between the nodes. The VC control information graph consists of the set of nodes; each of these nodes is attributed with the operation F_i. If two nodes q_i and q_{i+1} are connected with an arc, then the result of the operation F_i is the input for the operation F_{i+1}. Each node has an arc, which corresponds to the case when $F_i = 0$. Then VC is considered as being prohibited and no further analysis is performed.

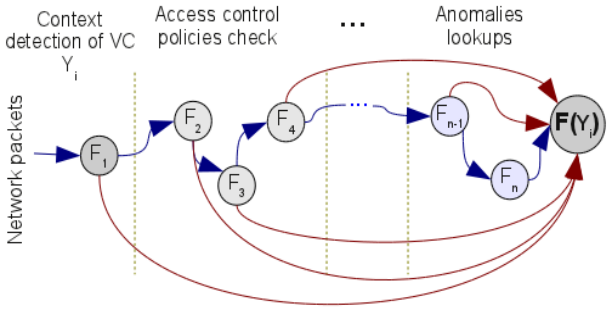

Figure 1. Virtual connection computation graph

The multiprocessor computing system which performs network traffic analyses might be described as a full mesh computation system graph with MIMD computers as its nodes. This graph is a full mesh, because the communications between CPUs are provided by hardware and operating system, and there is no predefined path between the cores; the data can pass directly from one node to another. Usually, the computation system graph and the control information graph do not match each other, because the amount of computing resources is limited and is less than the amount of computational processes. In this case, computation resources are used concurrently by information processes. It is possible to split the VC control graph in N non-crossing sub graphs and, thus, to build a VC operating pipeline. Because the virtual connections exist separately from each other, they can be processed in parallel. With the C compute nodes of MIMD type, the operating time of VC processing would be limited by (2).

$$T_{vc} = \frac{\max(z(f_i)) * \max(\tau_i)}{C} \quad (2)$$

Where $z(f_i)$ – number of CPU clocks, required for calculation of function f_i ,τ_i – average time of CPU clock in f_i calculation.

 ISBN: 978-1-61208-204-2

The given formula is an inequality because the decision on the VC classification (allowed/forbidden) can be made before passing all nodes of the graph.

Due to heterogeneity and re-configurability of the computing environments, in some cases the configuration of the firewall can be adapted to the access control tasks being solved at the current moment of time. This can be achieved by using the graph models for network traffic processing and Netgraph technology [4]. This technology allows organizing the network traffic processing in the context of the operating system [9].

Figure 1 shows an example of the virtual connections information control graph with decomposition of the indicator control function into components. The presented approach, in the combination with using the virtualization resources technology, allows improving performance of the network traffic monitoring and using only those computing components which are required for resolving the current access control problems.

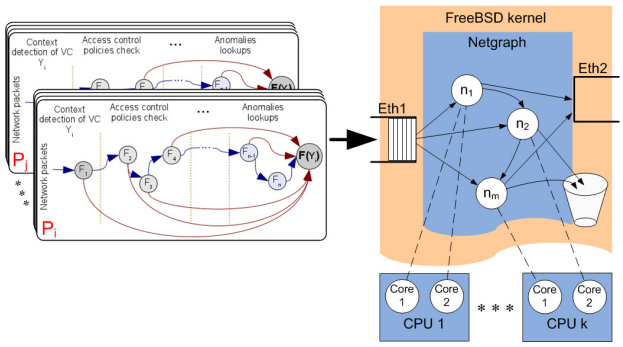

Figure 2. Information graph of the virtual connection management

Virtual connections should be processed on multiple cores. Network packets are balanced between cores using accessory to particular virtual connection. So, the order of packet flow in virtual connection is not corrupted, that allows process traffic in parallel using Netgraph network subsystem (Figure 2.).

III. VIRTUAL CONNECTIONS PROCCESING MODELS

To implement the parallel traffic processing Netgraph network subsystem, the part of FreeBSD kernel was used. This solution allows handling network connections in kernel mode and doing it using multiple cores. But the kernel mode programming produces new level of implementation complexity and delivers new behavior models which should be evaluated and carefully implemented. The cost of software bug is quite high; kernel level errors causes full system crash and reboot the firewall. But, if software stable are tested and verified, this approach will provide great performance opportunities. Well known Cisco software [10] also works in kernel mode and does it quite well. The graphs nature of Netgraph allows splitting traffic management process in independent parts logically and defining the computation process as a set of independent modules. The firewall configuration can be changed in runtime by adding and removing nodes in graph topology. It allows to extend firewall functionality and to improve performance by parallel traffic processing in separate kernel threads.

Netgraph has a complicated architecture and can operate differently, depending of the used nodes, the involved protocols and the implemented algorithms. When Netgraph starts, it creates a pool of kernel threads. The number of threads is equal to the number of available cores. These threads can be used for message processing. Network packets are presented as *mbuf* structures which are transferred between Netgraph nodes. In general, Netgraph can work in two modes – direct routine calls and queuing packets in nodes, and processing in multiple threads if possible. The operational mode depends on graph topology and node implementation. One of the reasons is function call depth. Recursive calls depth is limited by stack size. FreeBSD kernel stack is just 8K on i386 and 16K on amd64. It means that you can't pass more then 5-10 nodes without queuing (number of nodes depends on how much stack these nodes consume). There are two models which describe these Netgraph modes.

A. *Network driver based balancing and direct calls*

This solution fully depends on network interface kernel module implementation and used hardware. Not all network interfaces can handle traffic using multiple cores. Usually, it is implemented in high performance 10Gbit network interfaces. One of the possible technologies is MSI-X [11]. In this case, Netgraph uses direct calls to handle traffic. Traffic filtering works directly in network card – packet arrived event interrupts thread context. Netgraph uses the algorithm based on virtual connection attributes for balancing. For TCP connections it sends a packet of specific virtual connection to specific core. For UDP protocols it sends a packet to any available core.

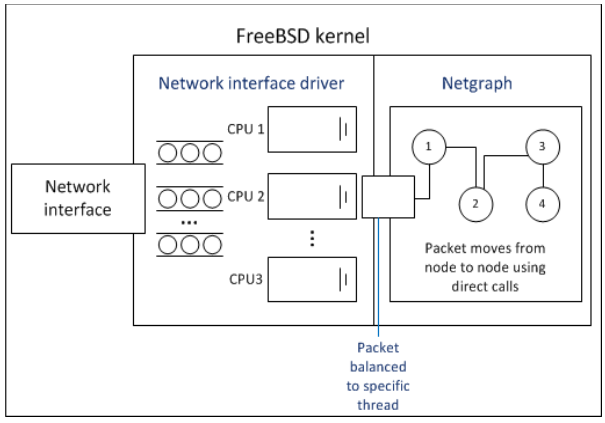

Figure 3. Load balancing in network interface driver

Figure 3 shows the structure of traffic processing in the FreeBSD kernel. A Packet arrives to network interface, then it gets processed by Netgraph Ethernet node *ng_eth* and

passes packet to the next module using direct routine call. Each next node evaluates the packet according to the filtering rules, the network protocol state model, the packet content and if the packet which belongs to some virtual connection is considered as allowed then the packet is sent to outgoing interface. If the virtual connection is prohibited, then the packet is dropped and the virtual connection is marked as prohibited. In this case, the traffic is routed through the Netgraph nodes, but it is processed in one thread. This behavior can could be presented as one process P which can be described as the process graph with the set of the states {s} and the set of the actions act(P):

$$S = \{s_i, i = 1..n\} \bigcup \{allowed, trash, wait\} \quad (3)$$

$$act(P) = \{p!.p?, d!, a!, b\} \quad (4)$$

where act(p) is the alphabet of actions, *p?* is incoming packet object, *p!* is outgoing packet object, *d!* is outgoing *drop* action (connection is prohibited), a*!* is outgoing *allowed* action – connection is allowed, no further analyses needed, b is *packet processed* action, system goes to accept the next packet. The process graph is shown on Figure 4.

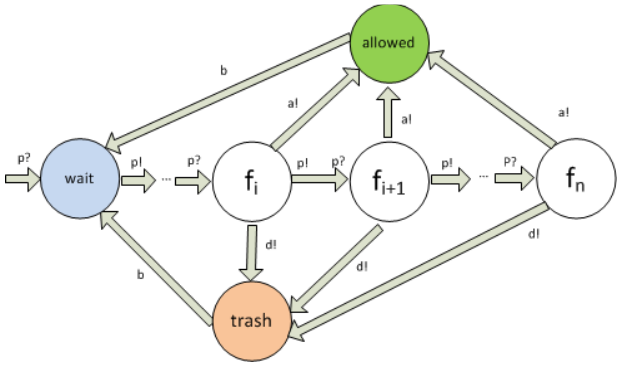

Figure 4. Connection control process graph

Process is awaiting for incoming event, when packet is scheduled to a specific thread it is evaluated by chain of Netgraph nodes, node accepts packet object, evaluates it and might generate three actions – decision is not made (to process packet on next node), connection is allowed, connection is prohibited (to move packet to trash).

B. *Queuing packets in nodes*

The second approach is to put packets in queues and process packets in Netgraph nodes in different kernel threads (Figure 5). In the described situation each Netgraph node is a separate process P_i, which can accept action messages with network packet object and produce the same messages as shown on Figure 4. But, the whole connection control process P is a parallel composition of Netgraph nodes processes:

$$P = (P_1 \mid P_2 \mid ... \mid P_n) \quad (5)$$

This solution allows to implement the parallel traffic processing using conveyers of nodes, which processes data in separate threads. The strong side of this solution is lack of hardware and network.

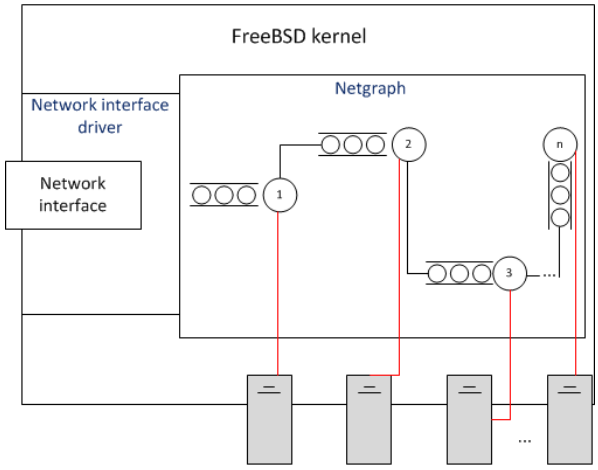

Figure 5. Netgraph nodes with queues

Figure 5 describes Netgraph behavior. Each node is processed on CPU or core in separate thread. When packet is arrived to node it put to FIFO queue.

IV. PERFORMANCE MEASUREMENT FOR NETGRAPH FIREWALL

We performed the experiments with firewall traffic control performance using Netgraph network subsystem. The first test is a scalability check. The virtualization technology was used in order to perform the experiment. The firewall was running as a virtual machine in Xen Cloud Platform hypervisor. The virtual firewall had two interfaces, which were connected to physical network using bridges in hypervisor service console. Figure 6 is the experiment schema.

 ISBN: 978-1-61208-204-2

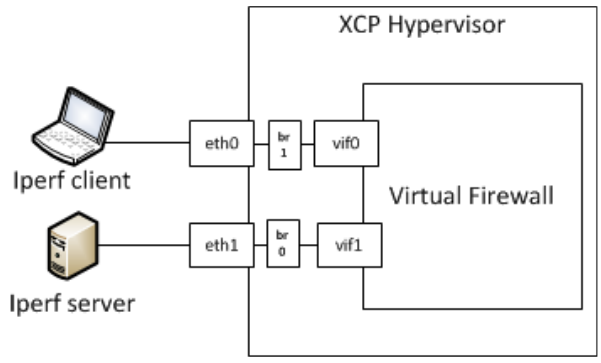

Figure 6. Scalability test experiment

We manually *slowed down* traffic filtering process in order to see how it scaled with the cores number growth. For performance measurement *iperf* tool was used in different configurations. Netgraph packet filter with packet queuing direct calls was used. Several TCP connections were created using *iperf* and this experiment was performed for one, two, and four cores configuration. Firewall performance is scaled almost linearly. The results are shown in Table 1.

TABLE I. TABLE 1. TRAFFIC CONTROL SCALABILITY TEST

	Direct calls model			**Packet queuing model**		
	1 TCP stream	2 TCP streams	4 TCP streams	1 TCP stream	2 TCP streams	4 TCP streams
1 core	1.43Mbit/s	1.43Mbit/s	1.43 Mbit/s	1.42 Mbit/s	1.44 Mbit/s	1.43 Mbit/s
2 cores	2.44 Mbit/s	2.45 Mbit/s	2.43 Mbit/s	1.52 Mbit/s	3.17 Mbit/s	3.14 Mbit/s
4 cores	2.44 Mbit/s	2.45 Mbit/s	2.44 Mbit/s	1.51 Mbit/s	3.18 Mbit/s	6.26 Mbit/s

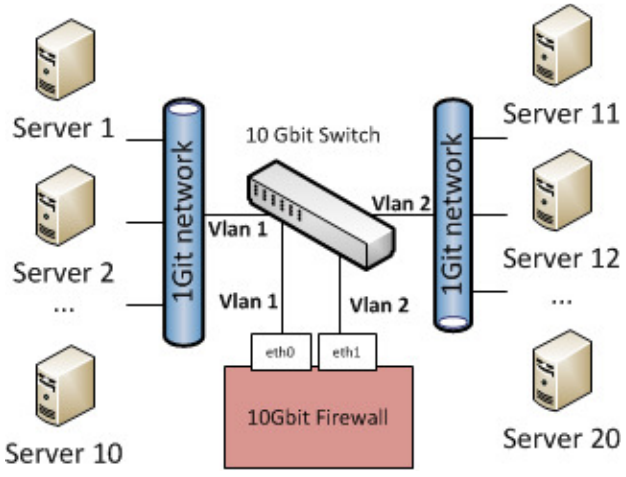

Figure 7. Performance test experiment

The second test in high performance environment was performed. Multicore system with two 10Gbit Ethernet adapters has been prepared. Firewall software has been installed on bare metal without virtualization. One gigabit Computer network with 20 hosts was separated into two segments with different VLANs and these segments were connected by stealth Netgraph firewall. Experiment schema is shown on Figure 7. Network hosts were configured to run iperf tests and generated the sufficient amount of network traffic. For this test Netgraph configuration with direct calls was selected, because hardware had 10Gbit network cards with MSI-X technology, so, traffic management was paralleled by network driver. We compared the kernel mode Netgraph firewall with the user space stealth implementation which uses Berkley Packet Filter (BPF) for traffic processing. The filtering algorithm is the same for both firewalls. The Experiment results are shown in Table 2.

TABLE II. TRAFFIC CONTROL ON MULTIPLE CORES

	Netgraph firewall	**BPF firewall**
Throughput, Gbit/s	8.3	1.2

V. VIRTUAL FIREWALL APPLIANCE IN THE CLOUD COMPUTING ENVIRONMENT

Information security in the cloud is a hot topic today [5]. There are no standards implemented in this area, but a lot of ideas were proposed. One of the major issues in virtualized systems security is an access control between virtual machines. Virtual machines communicate using network bridges in host system. Network bridge is implemented in Linux kernel and supports 802.1d standard. It can also be replaced by open vSwitch which supports more features like open flow, vlans, QoS, or proprietary bridge drivers, such as VMware vSwitch or Nexus 1000V. The paper proposes a solution which allows controlling traffic between virtual machines and having central management system. A typical distributed computing environment (cloud system) consists of the following software and hardware components:

- Virtualization nodes;
- Storage of virtual machines and user data;
- Cluster controller;
- Cloud controller.

Cloud computing systems might be used for the wide area of problems- from web services hosting government infrastructure and scientific computing. In Saint-Petersburg State Polytechnical University scientific cloud system based on OpenStack and Xen hypervisor was implemented. The distributed computing environment intended for solving scientific and engineering problems is a set of various computing resources such as virtual machines, and it has the following features [12]:

- The environment is used by a wide range of users, who are solving the problems of different classes;
- Virtual machines of different user groups can operate within one hypervisor;

 ISBN: 978-1-61208-204-2

- A wide range of software components (CAD/CAE applications, development tools) and operating systems (Linux, Windows, FreeBSD) are used;
- Different hardware configurations are used.

There is a difference in information security aspects between classic computing infrastructure, such as networks with hardware servers and user stations and virtual cloud environment where all resources are placed in the cloud, the hardware resources are shared between different users (possibly with different access rights):

- Information processing takes place on the virtual machines under full hypervisor's control; the hypervisor has access to all data processed by its virtual machines;
- Cloud software controls the resource planning and provision; it is a new entity in the information environment which has to be protected from the information security threats;
- Traditional information security components, such as hardware firewalls cannot control the internal virtual traffic between virtual machines in one hypervisor;
- In virtualized environments, files serve as virtual storage devices; these files are located in the network storages and are more exposed to threats than to hard disks;
- Transfer of instance memory occurs when virtual machines migrate between hypervisors; this memory may contain confidential information.

These features lead to the specific issues of security policy and access control in cloud systems. The environment becomes more dynamic. When the new resource (e.g. virtual machine) started in the cloud the security policy can be changed in the particular hypervisor or in the whole cloud system. For example, new virtual machine from security group "Engineering Department" was started. It changed the set of security groups in the particular hypervisor. So, the set of security policy rules was changed as well. That means, it is necessary to change the filtering rules for firewall dynamically. It controls network traffic between virtual machines, public network and other cloud components. Cloud computing system consists of virtualization nodes and cloud management services. Virtualization node is the hypervisor software which running on powerful multicore computing node. The domain level 0 (dom0 in terms of hypervisor XEN or service console in terms of other hypervisors) and virtual computing machines (domain level U, domU) operate in virtualization.

For information security and access control (AC) between the virtual machines that operate under a single hypervisor, the internal ("virtual") traffic and the external traffic (incoming from other hypervisors and from public networks) must be controlled. The solution of the access control problem could be achieved through the integration of a virtual firewall into the hypervisor; this firewall would function under the hypervisor, but separately from the user virtual machines. The virtual firewall domain can be defined as "security domain" (domS). Invisible traffic filtering is an important aspect of the network monitoring; the firewall must not change the topology of the hypervisor network subsystem. This can be achieved by using "Stealth" technology [13]; a packet traffic control is invisible to other network components**.** Virtual nature of firewall allows making hardware configuration dynamic. If security policy provides a lot of filtering rules, the number of the involved cores and memory amount can be dynamically increased. And vice versa, if the virtual firewall is not overloaded, it is possible to decrease allocated resources.

Figure 8 shows the common architecture of a distributed cloud system with integrated AC components. Abbreviations: VM – virtual machine; domS – security domain, virtual firewall; CSMS – the central security management system. The CSMS central management system generates and distributes the access control policies to all firewalls in the system. The security domain isolates virtual machines from the hypervisor, which prevents the possibility of attack against the hypervisor inside the cloud.

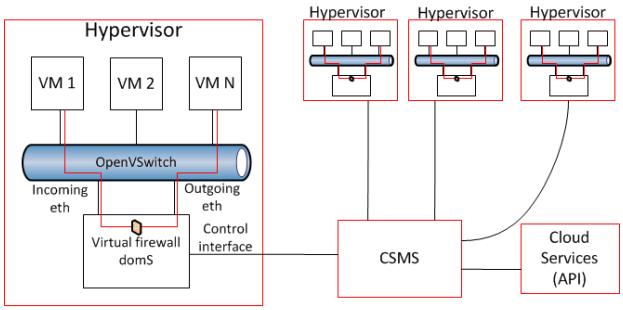

Figure 8. Secure cloud architectrure

Multicore stealth firewall based on Netgraph to implement traffic control between virtual machines was used. Virtual firewall has three network interfaces: two- for filtering and one- for management. Filtering interfaces are connected to open vSwitch bridge. Using OpenFlow technology channel level, the traffic routes were changed from the standard commutation tables – all outgoing virtual traffic routed to the incoming firewall interfaces. The firewall evaluates traffic in the stealth mode and passes it to outgoing network interface if it is allowed. From outgoing filter the interface traffic routed in a normal way by using commutation tables.

VI. CONCLUSION AND FUTURE WORK

The paper proposed the parallel traffic control model for high performance firewalls and describes firewall prototype implementation based on Netgraph network subsystem. The presented multicore firewall prototype shows good performance, up to 8.3 Gbit/s in 10Gbit networks. We also evaluated that solution based on graph model has good scalability. The firewall works in stealth mode and has not physical addresses and might be integrated to existing network topology without any changes. The firewall

 ISBN: 978-1-61208-204-2

software was tested in bare metal and virtualized environments.

The traffic management model with network card balancing requires hardware and software support side (at least, MSI-X technology), so it cannot be used in all systems. The second model (Netgraph nodes with queues) should work in all systems and we propose it as preferable. But implementation process of this model is more complicated and should be evaluated very carefully. The model with queues provides more control of traffic management. It allows performing load balancing by protocol types including nested protocols, e.g. MSI-X technology cannot perform load balancing for PPP protocol – all PPP connections are processed in single thread because it is treated as one virtual connection. Node queues also allow to override the existing Netgraph queue algorithm and to implement priority queuing as described.

Stealth mode allows implementing the information protection system for cloud computing environment in the form of a dedicated security domain (domS). The security domain can be quickly adapted to the current situation in the network and scaled if necessary because of firewall's virtual nature.

Described above architecture of secure cloud can be merged easily with low level methods of network control, for example, with flow-based traffic measurement or packet priority queuing management. The prototype of the described secure cloud environment based on OpenStack and adopted for CAD/CAE computation tasks, was created and currently in testing at the Telematics Department of the Saint-Petersburg Polytechnical University.

The future plan is to extend current virtual firewall prototype functionality. The prototype has to be adapted for work in different virtual environments such as VMware ESXi, Xen, Xen Cloud Platform, and KVM. The process models should be extended according to communicating sequential processes (CSP) theory and carefully checked because of potentially dangerous kernel operational mode.

REFERENCES

[1] Overview of Eucalyptus, 2011. URL: http://support.rightscale.com/09-Clouds/Eucalyptus/01-Overview_of_Eucalyptus 05.06.2012

[2] OpenStack: An Overview, 2012. URL: http://www.openstack.org/downloads/openstack-overview-datasheet.pdf 05.06.2012

[3] About the OpenNebula Technology, 2012. URL: http://opennebula.org/about:technology 05.06.2012

[4] Cobbs A., 2003. All about Netgraph URL: http://www.daemonnews.org/200003/netgraph.html 05.06.2012

[5] Cloud Security Alliance, Top Threats to Cloud Computing, 2010. URL: http://www.cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf 05.06.2012

[6] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. 2010. A view of cloud computing. Commun. ACM 53, 4 (April 2010), pp. 50-58.

[7] A. Silinenko. Access control in IP networks based on virtual connection state models: PhD. Thesis 05.13.19: / SPbSPU, Russia, 2010

[8] V. Zaborovsky, V. Mulukha. Access Control in a Form of Active Queuing Management in Congested Network Environment // Proceedings of the Tenth International Conference on Networks, ICN 2011, pp. 12-17

[9] Zaborovsky V., Lukashin A., Kupreenko S., 2010. Multicore platform for high performance firewalls. High performance systems // Materials of VII International conference – Taganrog, Russia.

[10] Vijay Bollapragada, Russ White, and Curtis Murphy. Inside Cisco IOS Software Architecture // Cisco Press 2008. 240 pages.

[11] Improving Network Performance in Multi-Core Systems http://www.intel.com/content/www/us/en/ethernet-controllers/improving-network-performance-in-multi-core-systems-paper.html

[12] Alexey Lukashin, Vladimir Zaborovsky, and Sergey Kupreenko. Access Isolation Mechanism Based On Virtual Connection Management In Cloud Systems // 13th International Conference on Enterprise Information Systems (ICEIS 2011), pp. 371 – 375

[13] V. Zaborovsky, V. Mulukha, A. Silinenko, and S. Kupreenko. Dynamic Firewall Configuration: Security System Architecture and Algebra of the Filtering Rules // Proceedings of The Third International Conference on Evolving Internet – INTERNET 2011, June 19-24, 2011, Luxembourg City, Luxembourg, pp. 40-45

 ISBN: 978-1-61208-204-2