Smart Systems and Devices

F. Derbel, N. Derbel, O. Kanoun,
R. Tourki, H.-R. Tränkler
(Editors)

SOGIC, Sfax, 2001

Parallel and Fault Tolerant Database for Access Control in Active Networks

Akbar[1], Buhari[1], Sahalu[2], Saleem[1]
[1]Lecturer, King Fahd University of Petroleum and Minerals.
[2]Instructor, King Fahd University of Petroleum and Minerals,
31261 Dhahran, Saudi Arabia.
Email: mibuhari@ccse.kfupm.edu.sa

Abstract

The Active Network paradigm allows programs to be injected into network routers so that customised user computations can be performed inside the routers in addition to their traditional packet forwarding fuction. An alternative implementation of active networks attaches to the transmitted packets the programs to be executed at the routers. A program residing in a router is activated only after verifying the packet that carries the reference to the code.

Routers normally use access control lists to process packets headers in order to make permit or denial of service to the packets. Processing incoming packets using the access control rules is time consuming and can degrade overall network performance. Traditionally, the rules in the access control lists are interpreted sequentially to make decision on packet forwarding.

In this paper we propose a mechanism for processing the rules in the access control list in parallel in order to improve network performance. Our system achieves fault tolerance by using Java threads. Early experience with our systems indicates that it can improve network performance by up to 20%

Keywords: Applet oriented Active Network, Parallel Processing, JDBC

1. Introduction

The emergence of Active Network technology has attracted many academics and researches to become involved in the development of this technology. Active network provides opportunities for the Information Technology world due to several factors. First, active network technology can be used to reduce the time required for the standardization process of new network services. Second, active network shifts the conventional network paradigm from a passive node that only transfers bits to a more general processing engine like an end station. Furthermore, it would also make the possiblity of enabling on-the-fly modification of network functionality, for example to adapt to changes in link conditions.

The infusion of Java technologies has enabled the Internet industry to create dynamic, content driven, platform independent and distributed Internet applications. Using Java applets and the industry standard Java Database Connectivity API (JDBC), it is now possible to built powerful Internet based database applications with sophisticated graphics user interface and interactive database access, comparable to their native operating system and database counterparts. Our active network implementation is based on the use of java technology and a parallel system which helps to do processing faster as well as providing some support for fault tolerance (using Threads).

Takeshi et. al.[9] have proposed a compiler for parallelizing IP-packet filter rules, to improve the network security and reduce the degradation in packet-forwarding performance. Maruyama et al[11] describe a hight-speed IP router from the points of view of parallel processing granularity and dedicated processing. Our approach to efficient routing protocol processing and packet-forwarding is based on the use of parallel active networks.

The basic idea behind this system is that when a sender wishes to send a request to a receiver, the permission for the required transmission has to be checked using an Access Control List (ACL). The rules are checked sequentially and this can lead to network performance degradation. We have developed a java-based active network implementation which uses PVM to process the ACL rules in parallel. We report the design and our experience with this system.

The rest of this paper is organised as follows: Section 2 gives an overview of active networks in general and our active network model in particular. In section 3 we outline the issues of network access and firewall design policy. A brief overview of access control list is presented in Section 4 and in Section 5 we describe the process of using a java applet to access the server in our implementation. The operation of our proposed model is discussed in Section 6. We discuss the *pros* and *cons* of our system in Section 7. We discuss our preliminary experimental results in Section 8 and conclude the paper in Section 9.

2. System description

2.1. Types of active network system

There are two possible approaches to build active networks. A discrete or out-of-band approach and an integrated or in-band approach [Raj97].

2.1.1. The Discrete Approach: This may also be called a *programmable node* (switch/router) approach. Here programs are injected into the programmable active node separately from the actual data packets that flow through the network. A network user would send the program to the network node (switch or router), where it would be stored and later executed when the data arrives at the node, processing that data. The data can have some information that would enable the node decide how to handle it or what program to execute.

2.1.2. The Integrated Approach: In an integrated approach, also termed as the *encapsulation approach*, the program is integrated into every packet of data sent into the network. These packets that carry programs in addition are called *capsules* in the literature to distinguish them from passive data-carrying packets. When these capsules arrive at the active node, it interprets the programs and sends the embedded data depending on its interpretation of these programs. This concept is similar to Postscript code, where actual data is embedded in program fragments that the printer understands. In this approach, each active node would have built-in a mechanism to load the encapsulated code, an execution environment to execute the code and a relatively permanent storage where capsules would retrieve or store information.

ACL is a database of rules that the router consults in order to make packet forwarding decisions. In existing systems the access control

2.2. Our active network system

Our system is based on the programmable node approach whereby programs are injected into the network using separate packets than those that carry data. We prefer for our active network platform that the packet carries only identifiers or references to the services that reside in the node due to the following reasons:

1. It maintains the packet size to minimum
2. The idea is to optimize the resources available in every machine connected to a LAN. Code can be stored in non-exhaustive machines.
3. The packet format can be designed in conformance with the Parallel processing architecture of interest.

For initial experimentation, we have setup an active network system with three computers. Two computers act as an end system and another computer is treated as a router. The configuration is shown in **Figure 1**.

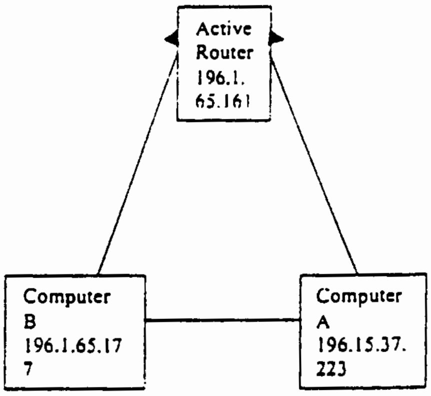

Figure 1. The Configuration Scenario

Although the machines are physically connected through the Ethernet, we design that all communication should be made through the router. PVM resides in the active router and the other two computers (Computer A and Computer B) act as PVM daemon.

3. Network services and firewall policies

A key decision that ought to be taken when developing the security policy of a computer network is the design of firewalls. A firewall is a hardware or software barrier placed between the network of concern and the rest of the world to prevent unwanted and (potentially) damaging intrusions into the network.

1. Network Service Access Policy: A higher-level, issue-specific policy which defines those services that will be allowed or explicitly denied from the restricted network, plus the way in which these services will be used, and the conditions for exceptions to this policy. When restricting a service, network service access policy should also include all other outside network access such as dial-in and SLIP/PPP connections.
   It is better to draft the policy before the firewall is implemented. While protecting the resources, there should be ample ways to access the outside world. Special rights for certain users should also be provided.

2. Firewall Design Policy: A lower-level policy which describes how the firewall will actually go about restricting access and filtering the services as defined in the network service access policy. The major concept here is based on two policies:
   i. Permit any service unless it is expressly denied; or
   ii. Deny any service unless it is expressly permitted.

The second policy follows the classic access model used in all areas of information security.

4. Access Control List

Firewalls are highly needed to secure the system from an intruder's access. The firewall has to use an IP router to control the passing of any packet from the Internet into the Intranet. The packet filtering process involves parsing the headers of the received IP packets and forwarding or discarding the packets according to the Access Control Rules specified by a network administrator. The performance of the whole router depends on the procedure on which the rules in the Access Control List (ACL) are applied to the packet and a decision is made to allow or reject the packet. ACL's are the lists present in the routers which contain the conditions that ought to be satisfied for permit or denial of entry into a network. Handling all the incoming packets using the access control rules consumes a lossof time and so it degrades the total performance of a network. Also, the performance of the off-the-shelf routers is degraded in proportion to the complexity of the ACL. Hence IP routers, that forward packets more efficiently, are needed.

The conventional method of packet filtering is achieved by sequentially interpreting the ACL, to determine whether a packet should be forwarded or discarded. The access-control decision facility itself is now generally conceived to be based on a system of access-control rules in which specific policies are expressed. The rules in turn generally depend on properties of the system context (e.g., the time), the target object (content-dependent access control), and the requester (user-dependent-access control). An ACL, specified by the network administrator, is a set of conditions the packets must satisfy and actions to be performed upon the packet and is shown in Table 1. The process of filtering a packet is done by checking a set of six input parameters. These parameters are the Packet type, the Source IP address, the Source TCP/UDP port, the Destination IP address, the Destination TCP/UDP port and the Acknowledgment (ACK) bit. The output is the action to be performed. An action has a value either to "Permit" or "Deny": "Permit" indicates that a packet should be forwarded and "Deny" indicates that it should be discarded. The action specified in a row is considered only when an input pattern matches the respective components of the corresponding row.

Conditions(C)						Action(A)
Packet Type	Source IP Address	Source TCP/UDP Port	Destination IP Address	Destination TCP/UDP Port	ACK Bit	

Table 1. Format of Access Control List

Traditionally, packet filtering is done using sequential parsing methods [9]. With the sequential parsing technique, increase in the number of rules in the ACL list will cause the parser to consume more time and become inefficient.

Sequential parsing causes the following problems [10,9]:

1. As the number of rows of the rules in the ACL increases, the cost of packet filtering also increases.
2. Because a condition consists of conjunctions of parameters, disjunctive conditions must be specified in several rules. In these rules, each kind of parameter has the same value unless it specifies a disjunctive value. As a result, the same value might be applied to a packet many times.
3. Consider the conditions of the i^{th} rule and the k^{th} rule in the ACL, where i < k. If the Condition of the i^{th} rule is always true when the Condition of the k^{h} rule is true, the rule in the k^{h} row of the ACL is redundant because its action, A_k, Is never carried out. This is similar to an infeasible path (IFP) in a

procedural-program and this has to be remedied...

Takeshi et. al.[9] have proposed a compiler for parallelizing IP-packet filter rules, to improve the network security and reduce the degradation in packet-forwarding performance. Maruyama et al[11] describe a hight-speed IP router fromthe points of view of parallel processing granularity and dedicated processing.Our approach to efficient routing protocol processing and packet-forwarding is based on the use of parallel active networks.

5. Applet to Server access

The JDBC API, categorizes JDBC drivers into four types: 1) JDBC-ODBC bridge, 2) native-API partly-Java driver, 3) net-protocol all-Java driver, 4) native-protocol all-Java driver.

We are using a driver type that belongs to category 3. The first and second types are of the JDBC drivers which are not fully compatible with applet oriented connections. The net-protocol all-java driver is described as, "A *net-protocol all-Java driver* translates JDBC calls into a DBMS-independent net protocol which is then translated to a DBMS protocol by a server. This net server middleware is able to connect its all Java clients to many different databases. The specific protocol used depends on the vendor. In general, this is the most flexible JDBC alternative. It is likely that all vendors of this solution will provide products suitable for Intranet use. In order for these products to also support Internet access they must handle the additional requirements for security, access through firewalls, etc., that the Web imposes. Several vendors are adding JDBC drivers to their existing database middleware products."

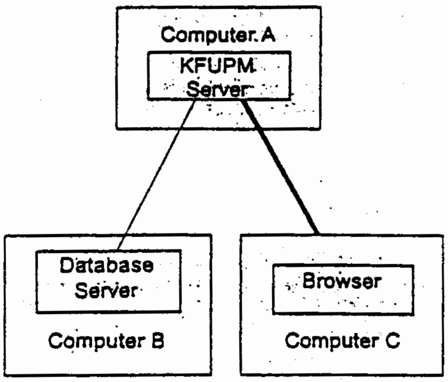

Figure 2. Net server middleware architecture

6. Steps for our system processing

From Client side:

1. The client's IP address is found out automatically using Java methods
2. The client can now select the destination to which to send the packet, among the exisiting destinations list. This list is maintained because we are currently experimenting with the system within a subnet.
3. The client, then selects the protocol which is desired to be used to send the packet. The protocol options are TCP, UDP, IPv4 and IPv6.
4. The user is allowed to select a list of processes which he wishes to execute at the router level or at the destination. Available programs that can be selected at this point in the implementation are Matlab and PVM.
5. Then the user can start sending data.

From Server Side:

1. The appropriate server for the respective protocol can be started.
2. There are options for multicasting also.

Processing done:

1. After the client gives the send request, the JDBC is called to access the database (MS-Access) to determine the permission provided. If the permission is present, then the process is allowed. If not, the packet transmission is denied. As the JDBC cannot talk to the server, we use the net-server middleware called as IDS.
2. Then, the routing table is accessed to determine to which node the packet has to be transmitted if the source and destination is known.
3. To access the database and to change the ACL, there is a password setup for the database, without which the user cannot access the database.

Database maintainence:

1. The processing can be done at the Database in parallel depending upon the position of the client and the current utilisation of the network.
2. The updation of the Databases are done immediately. When the administrator performs an update on one database, then there is a script used to update on other databases with the lock being done at the database retrival.

7. Advantages and limitations of our system:

7.1. Advantages:

1. The system does the processing concurrently on the database, if more than one request occurs at the same time. Conceptually, a single Connection instance can service multiple concurrently executing statements, PreparedStatement (the statements that are ready and have the resources to be exectued) and CallableStatement (the statement which direct any function call) instances. It is quite natural to come up with a Java program in which multiple query Statements are created from a single Connection. Then these Statement objects are used by different threads, or their repeated executions are interweaved with each other.

   However, the underlying database system may not support this kind of operation. Note that the IDS Server does not change the capability of the database system in this manner. You can find out whether this feature is supported by calling DatabaseMetaData.*getMaxStatements*(). A return value of 0 means the database system supports unlimited number of concurrently executed Statements created from a single Connection. Returning 1 means only one active Statement per Connection is allowed, or you need a separate Connection for another active Statement.

2. The processing can be done on-the-fly with the help of PVM and Matlab. The router can distribute the job using PVM and PVMD. Various processings can be done using Matlab.

3. The whole system is heterogenous. Currently, it works on Windows NT, Windows 98 and Linux.

7.2 Limitations:

1. The system only supports Netscape Navigator but not Internet Explorer.
2. The system is currently being tested only within an Intranet.

8. Preliminary results

Initial experiments were carried out when in our system when the control rules are specified in a text file and the program just accesses the text file to access the data inside the file.

Number of rules present: 64.
Connectivity time: 0 msec
Time to access the first rule: 10msec
Time to access the middle or last rule: 20msec

Testing the database access in the system was performed under the following criteria:

1. When the client program, the IDS server and the Database exists on the same machine [Pentium 200 MHz, 64MB RAM].
2. When the client and IDS server are present in one machine [Pentium 200 MHz, 64 MB RAM] and the Database exists on a different machine [Pentium 100 MHz, 32MB RAM].

Case 1:

Rule position	Connectivity time (msec)	Access time (msec)
1	1322	971
32	1776	2103
64	1252	921

Case 2

Rule position	Connectivity time (msec)	Access time (msec)
1	2013	1352
32	2323	3135
64	1512	3815

While the program was executed, it was found that the CPU usage has raised form 4% to 32% and the RAM usage has increased by 2%.

9. Conclusion

With the emergence of Active Networks, the network equipments have now become active instead of being passive. Now, by the introduction of parallel and fault tolerant approach on the database access while doing the access control processing by any router, the active processing of the network equipment has increased its performance. Further work has to be done on this side so as to make the system viable to Internet oriented applications as well.

10. References

[1] Calvert, K.L, Architectural Framework for Active Network version 1.0, Active Network Working Group RFC, University of Kentucky, July 27, 1999

[2] Bhattacharjee, S; Calvert, K.L.; Zegura, E.W., Implementation of an Active Networking Architecture, Technical Report, Georgia Institute of Technology, July 1996

[3] Bhattacharjee, S; Calvert, K.L.; Zegura, E.W., An Architecture for Active Networking, Proc. IEEE INFOCOM '97, April 1997

[4] Wetherall, David J; Guttag, J. V.; Tennenhouse, David L, ANTS: A Toolkit for Building and Dynamically Deploying Network . Protocols, Proc. IEEE OPENARCH '98, April 1.998

[5] Schwartz, Beverly; Zhou, Wen Yi; Jackson, Alden.W, SmartPackets for Active Networks, BBN Technologies, Jan. 1998

[6] Alexander, Scott D., *et al.*, Active Network Encapsulation Protocol(ANEP), RFC Draft, Juli, 1997

[7] Banch, Albert; *et al*, Multicasting Multimedia Stream with Active Networks, ICSI Technical Report 97-050

[8] Wilkinson, B., Allen, W., Parallel Programming: Techniques and Applications using Networked Workstations and Parallel computers, Prentice-Hall Inc., 1999.

[9] Takeshi Miei, Mitsuru Maruyama, Tsuyoshi Ogura & Naohisa Takahashi (1997). Parallelization of IP-Packet Filter Rules. Proceedings of Third International Conference on Algorithms and Architectures for Parallel Processing '91, 381-388.

[10] Mansour Esmaili, Rei Safavi-Naini, Bala Balachandran & Josef Pieprzyk (1996). Case-Based Reasoning for Intrusion Detection. Twelfth Annual Computer Security Applications Conference '96, 214-223.

[11] Mitsuru Maruyama, et al. COURrouter-I: An Experimental Parall IP Routr Using a Cluster of Workstations. IEICE TRANS. COMMUN., Vol. E80B, No. 10 October 1997.