Int. J. Communications, Network and System Sciences, **2011, 4, 683-695**
doi:10.4236/ijcns.2011.411084 Published Online November 2011 (http://www.SciRP.org/journal/ijcns)

A Topology-Based Conflict Detection System for Firewall Policies Using Bit-Vector-Based Spatial Calculus

Subana Thanasegaran, Yi Yin, Yuichiro Tateiwa, Yoshiaki Katayama, Naohisa Takahashi
Department of Computer Science and Engineering, *Nagoya Institute of Technology*, *Nagoya*, *Japan*
E-mail: {*subana*, *katayama*, *tateiwa*, *naohisa*}*@moss.elcom.nitech.ac.jp*, *yi*837*@hotmail.com*
Received September 1, 2011; *revised September* 30, 2011; *accepted October* 12, 2011

Abstract

Firewalls use packet filtering to either accept or deny packets on the basis of a set of predefined rules called filters. The firewall forms the initial layer of defense and protects the network from unauthorized access. However, maintaining firewall policies is always an error prone task, because the policies are highly complex. Conflict is a misconfiguration that occurs when a packet matches two or more filters. The occurrence of conflicts in a firewall policy makes the filters either redundant or shadowed, and as a result, the network does not reflect the actual configuration of the firewall policy. Hence, it is necessary to detect conflicts to keep the filters meaningful. Even though geometry-based conflict detection provides an exhaustive method for error classification, when the number of filters and headers increases, the demands on memory and computation time increase. To solve these two issues, we make two main contributions. First, we propose a topology-based conflict detection system that computes the topological relationship of the filters to detect the conflicts. Second, we propose a systematic implementation method called BISCAL (a bit-vector-based spatial calculus) to implement the proposed system and remove irrelevant data from the conflict detection computation. We perform a mathematical analysis as well as experimental evaluations and find that the amount of data needed for topology is only one-fourth of that needed for geometry.

Keywords: Packet Filtering, Misconfiguration, Network Security, Spatial Analysis

1. Introduction

A firewall protects the network from unauthorized access and provides secure access to the outside world. Packet filtering techniques in a firewall provide an initial level of security and operate on the network layer or the internet layer. Packet filtering controls the network traffic with a predefined, ordered set of filters called a firewall policy (FP). Every filter *f* in the FP has a condition and an action. If a filter matches a packet *P* the filter action is carried out on packet *P*. The action can be either accept or deny the packet. Many packet filtering schemes such as IPFW or IPFIREWALL in FreeBSD is a first matching filtering scheme [1]. IPFW is a FreeBSD sponsored firewall software application enables use of sophisticated filtering capabilities.

Managing and maintaining firewall policies is an extremely complicated job for the network administrator because the policies are constantly subject to modification. For example, while examining 37 firewalls in production enterprise networks, Wool found that all the firewalls were misconfigured and vulnerable [2].

When the network behavior is different from the actual configuration of a policy, potential security holes are created. For example, if an IP address is mistaken in an FP, it allows unintended traffic in the network and can cause security holes. A conflict is a type of misconfiguration that occurs when a packet P matches two or more filters. It is a common misconfiguration in firewalls. Hamed *et al.* found that there is a high probability of even expert system administrators and network practitioners creating conflicts [3]. Conflicts render the filters redundant or shadowed, and as a result, the actual configuration of the firewall policy is not reflected in the network.

Various techniques have been developed to manage firewall policies [3-22]. Among these are firewall analysis tools [4,5] and techniques that focus on minimizing the firewall rules [6]. Other related works detect conflicts by using geometrical analysis [7-9]. Yin *et al*. developed a conflict detection technique based on geometrical analysis, which provides systematic conflict classifica-

tion [8]. However, this technique becomes very demanding in terms of memory and computation time when the number of header fields and filters increases. To solve this problem, we have presented a topological approach to detect conflicts [10,11]. In this approach, we introduced a bit-vector based spatial calculus named BISCAL and constructed a new conflict detection framework through BISCAL which analyzes the topological relationship of the filters to detect and classify conflicts. This paper extends and strengthens that framework by describing the design and implementation of the topology-based conflict detection system that shows how effectively the BISCAL can be adopted for conflict detection and by presenting quantitative evaluations of the system. In this paper, our main contributions are as follows:

1) We have presented a systematic implementation method based on BISCAL to construct a conflict detection system that efficiently classifies conflicts by treating with topological relationships between filters effectively, and can remove the irrelevant data from the conflict detection in the intermediate stage of computation.

2) We have performed comparative studies through mathematical and experimental analysis to show that the topological approach requires much less computation time and storage than geometrical approach in conflict detection.

We implement the topological approach using BISCAL to extract the conflicting filters from the firewall policy. The main advantage of BISCAL is that the operations on the filter sets become easier. The rest of the paper is organized as follows. In Section 2, we discuss related works. In Section 3, we explain the firewall policy used in the conflict detection system. Section 4 discusses the spatial relationship of the filters, and Section 5 explains the conflict classification in detail. The operators and supporting vectors of BISCAL are explained in Section 6. Section 7 gives an overview of the proposed system and its computational steps. The evaluation and comparative analysis of our approach are discussed in Section 8. And finally, in Section 9, we present the conclusions and discuss future research directions.

2. Related Work

Over the past few years, research and analysis of firewall policies has received considerable attention [2-28]. Al-Shaer *et al.* proposed an algorithm to detect conflicts caused by one filter on another in a firewall policy on the basis of the relationship between the two filters [12]. This relationship was defined according to their predicates such that they satisfied the conditions in $\{\subset \supset =\}$. However, the problem with this algorithm is that when the corresponding predicates of two filters overlap, the relationship between the two filters cannot be determined, because overlapping predicates do not satisfy any of the conditions in $\{\subset \supset =\}$. Therefore, the algorithm for detecting conflicts also does not work when overlapping predicates appear between two filters.

V. Capretta *et al.* proposed a formalization of conflict detection for firewalls. They defined conflict for the rules if and only if the actions of the rules are different [13]. In such conflict detection, redundancy cannot be detected, as the redundant filters have the same action, but our system analyzes the filters with both the same and different actions, and conflicts are classified in detail.

Mayer *et al.* developed a firewall analysis tool Fang to perform customized queries on a set of filters and to extract the filters in a firewall policy [4]. Wool *et al.* improved the usability of Fang [5]. These tools and methods help administrators to manually verify the correctness of a firewall policy. Unfortunately, they require a high degree of user expertise to write proper queries and identify the problems in firewall policies.

H. G. Verizon *et al.* proposed a fast and scalable method for resolving the anomalies in firewall policies [14], which can be useful for large-scale firewall policies. H. Hu *et al.* proposed a firewall anomaly management and resolution environment: FAME, and developed a grid-based visualization of the firewall policy [15]. Liu *et al.* used firewall decision trees to detect and remove the redundant filters [16]. K. Matsuda proposed a model called matrix decomposition to analyze the filters in the firewall policy with a few compression methods [17]. If for some reason, an administrator needs to intentionally embed redundant filters, which will conflict with the other filters in a firewall policy, we will inform the administrator of the presence of conflicts instead of removing these redundant filters. This also helps the administrators to add, delete, or modify the existing filters according to their own intentions, as we have classified the conflict contents in detail.

T. Srinivasan *et al.* proposed a scalable and parallel packet classification method using an aggregated bit vector [24]. Lakshman *et al.* proposed high-speed policy-based packet forwarding using bit-level parallelism [25]. Both the above methods yield good results by using the bit-vector data structure for packet classification schemes. Baboescu *et al.* proposed a fast and scalable conflict detection technique for packet classifiers using a bit-vector. It detects the conflicting pairs, but it does not classify the conflicts explicitly [22]. Our method efficiently utilizes the bit-vector in our conflict detection system through BISCAL and explicitly classifies as errors and warnings, which helps administrators to re-configure the policies more easily.

3. Background

An FP consists of an ordered set of m filters, and it is expressed as follows:

$$\text{FP}:\left(f_0, f_1, \cdots, f_{m-1}\right)$$

where if two filters, f_i and $f_j\left(i, j \in [0, m-1] \text{and } i, j\right)$ are such that $i < j$, filter f_i is placed before f_j in the FP. We follow the first matching filter scheme, and therefore, f_i has a higher priority than f_j during execution. Each filter $f_i\left(i \in [0, n-1]\right)$ consists of n key fields, called predicates $p_{i0}, p_{i1} \cdots p_{in-1}$ and an action. The filter f_i is expressed as shown below:

$$f_i : p_{i0}, p_{i1} \cdots p_{in-1}, \text{ action}$$

The commonly used matching keys in a header are protocol (represented as Pro), source IP (SrcIP), destination IP (DesIP), source port (SrcPort), and destination port (DesPort).

Each predicate $p_{ik}\left(i \in [0, m-1], k \in [0, n-1]\right)$ can be represented as an exact value, a prefix, a range value, or a list in many firewall systems. However, in this paper, we use range value for the sake of simplicity and because a filter with predicates in other forms can be easily converted into one or multiple filters with range values. Each predicate $p_{ik}\left(i \in [0, m-1], k \in [0, n-1]\right)$ is represented as $a_{ik} \le u_k < b_{ik}$, by using a uniform range value $[a_{ik}, b_{ik})$ and the value in the k^{th} key field of the packet header, u_k. The default filter is the lowest priority filter that denies access to all the packets when no other filter matches the packet.

We represent a filter in the form called internal form, which includes the range values instead of the predicates in n key fields. An internal form filter f_i is represented as follows:

$$f_i : [a_{i0}, b_{i0}), [a_{i1}, b_{i1}) \cdots [a_{in-1}, b_{in-1}), \text{action}$$

Assume that the values of the header of packet P are represented as $u_0, u_1 \cdots u_{n-1}$. A packet P matches f_i if and only if $a_{i0} \le u_0 < b_{i0}, a_{i0} \le u_0 < b_{i0} \cdots a_{in-1} \le u_{n-1} < b_{in-1}$. An example policy consisting of internal form filters is shown in **Table 1**, where f_5 is the default filter.

4. Spatial Relationship of Filters

According to Max J. Egenhofer, we can classify the spatial relationships on the basis of the spatial concepts [29]. The three relationships are 1) topological relationships, 2) spatial and strict order relationships, and 3) metric (geometric) relationships. Here, we focus on geometric and topological relationships and analyze how they differ in terms of conflict detection.

Table 1. A sample firewall policy FP1.

	SrcIP	DesIP	Action
f_0	[0,3)	[3,7)	Accept
f_1	[0,3)	[3,7)	Deny
f_2	[1,5)	[1,5)	Accept
f_3	[3,5)	[1,3)	Deny
f_4	[5,7)	[1,3)	Accept
f_5	[0,2^{32})	[0,2^{32})	Deny

4.1. Geometric Relationship of Filters

Max J. Egenhofer stated that metric relationships are based on parameters such as distance and directions of the objects [29]. Finding the metrics of an object has many applications in the field of telecommunication, medical imaging, robot motion planning, etc. For example, multi-dimensional packet classification is viewed as *a point location problem* in **computational geometry**. Finding the geometric location of a packet in an n-dimensional space is necessary to classify a packet in packet classification techniques [23-26]. The n-dimensional space is otherwise called as **packet space**.

When the number of key fields in a packet is n, the packet can be represented as a point in n-dimensional space or packet space. A filter is represented as a subspace of a packet space, called the **filter space** FS, which includes all the points of the packet that match the filter. The geometric shape of the filters in a two-dimensional space is a rectangle, and in an n-dimensional space, it is a hypercube or n-cube. The sample firewall policy in **Table 1** is represented spatially in **Figure 1**.

We perform a spatial decomposition of the n-dimensional space until dividing the filter boundaries reaches its last dimension. The final decomposed space is called a subspace S_i. The packet classification techniques in [23-26] preserve the subspaces and their locations in the n-dimensional space in various data structures to classify the packet in n key fields. In packet classification, the location of a subspace in the n-dimensional space is essential for finding which filter matches which packet. However, conflict detection techniques that find conflicting filters preserve the same data as packet classification techniques [7-9]. D. Eppstein *et al.* used orthogonal range searching techniques and proposed techniques for the packet classification problem and the conflict detection problem simultaneously [7]. Yin *et al.* detected the conflicting filters that appear in [8] by dividing the filter space using SIERRA, a systolic filter sieve array used in high-speed packet classifiers [23].

Both the above techniques detect the conflicting filters based on packet classification analysis. However, when

the number of key fields and filters increases, the conventional methods used in [7-9] are extremely demanding in terms of memory and computation time, as they depend on geometry-based packet classification techniques. As far as conflicts are concerned, rather than using packet classification, we should focus on filter classification. Therefore, in our paper, we propose to focus only on filter classification and therefore solve the drawbacks of the previous approaches based on the geometrical relationship of the filters.

4.2. Topological Relationship of Filters

According to Max J. Egenhofer, topological relationships are a subset of spatial relationships [29]. Topological notations include the concepts of continuity, closure, interior, and boundary, and are defined in terms of neighborhood relations. Topological equivalence is a crucial criterion when comparing the relationships between objects. Topological equivalence does not preserve distances and directions. Topology refers to the way in which the filters are connected to each other. Therefore, filter classification can be performed by identifying the topological relationship of the filters, which is the basic requirement for conflict detection.

In our approach, as we have discussed earlier, we perform the *n*-dimensional spatial decomposition, and then further analyze the filters in the subspaces to find the topological relationships of the filters. The difference between the number of subspaces required for conflict detection using topology and geometry is explained below. Geometry-based systems consider all the subspaces for conflict detection, as the location of all the subspace is important for packet classification techniques. In a topology-based system, the location is discarded, and concentrates only on the uniqueness of the subspace. In other words, it only selects the subspaces with different filter sets by removing subspaces with the same filter sets.

Therefore, the number of subspaces (the amount of data required for conflict detection) is smaller for topology-based systems than for geometry-based systems.

For example, in **Figure 1**, the number of subspaces considered in the geometrical approach is thirteen. In the topological approach, it is five, because there are only five unique subspaces with different filter sets, namely $\{\{f_0, f_1\}, \{f_0, f_1, f_2\}, \{f_2\}, \{f_2, f_3\}, \{f_4\}\}$. As a result, when the number of filters and key fields increase, the difference in the number of subspaces considered for conflict detection varies drastically between the topological and geometrical approaches. Therefore, we can say that identifying only the topological relationships improves the efficiency of the conflict detection system by removing the irrelevant data from the conflict computation. In this way, we can achieve our target of solving the two issues, namely, large memory usage and computation time. We use experiments to verify this in later sections.

Conflict detection is performed by finding the topological relationship of the filters. There are nine possible topological relations between any two objects, as they appeared in [29]. We have extracted five topological relationships (TR) to identify conflicts between any pair of filters, f_i and f_j. In other words, TR (f_i, f_j) = {*disjoint*, *inside*, *contains*, *equal*, *overlap*}. The relations are shown in two dimensions in **Figure 2**, and can be generalized for higher dimensions. The filter space of a filter *f* is represented by FS (*f*).

Disjoint: $\mathrm{TR}(f_i, f_j)$ = disjoint when the intersection of the filters spaces is empty or $\mathrm{FS}(f_i) \cap \mathrm{FS}(f_j) = \varnothing$, as shown in **Figure 2(a)**.

Inside: $\mathrm{TR}(f_i, f_j)$ = inside when f_j is completely enclosed by f_i or $\mathrm{FS}(f_j) \subset \mathrm{FS}(f_i)$, as shown in **Figure 2(b)**.

Contains: When f_i is enclosed by filter f_j, or $\mathrm{FS}(f_j) \supset \mathrm{FS}(f_i)$, then we say that there exists a relation $\mathrm{TR}(f_i, f_j)$ = contains between filters f_i and f_j, as shown in **Figure 2(c)**. Contains is the converse of the inside relation.

Equal: $\mathrm{TR}(f_i, f_j)$ = equal when f_i and f_j are equal, or $\mathrm{FS}(f_i) = \mathrm{FS}(f_i)$, as shown in **Figure 2(d)**.

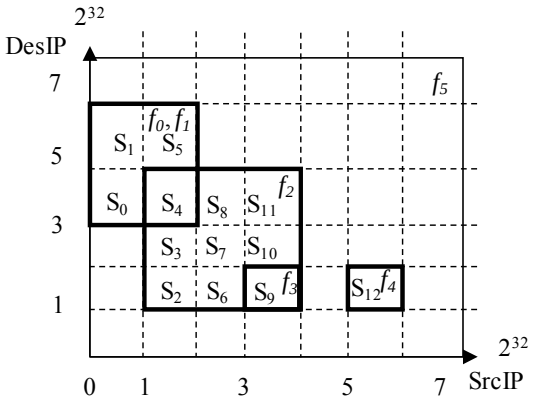


Figure 1. Spatial decomposition of filters of FP1 defined in Table 1.

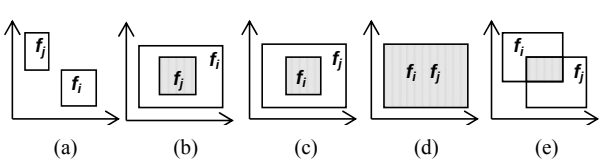


Figure 2. Topological Relationships. (a) Disjoint; (b) Inside; (c) Contains; (d) Equal; (e) Overlap.

Overlap: When f_i and f_j do not satisfy any one of the above four relationships, then we can say that $\mathrm{TR}(f_i, f_j)$ = overlap, as shown in **Figure 2(e)**.

5. Classification of Conflicts

Conflicts are classified in filter pairs, (f_i, f_j), where $i < j$ and filter f_j has an error or warning caused by f_i. The errors and warnings of the filter f_j can be classified exclusively by the type of topology between f_j and f_i and the actions of f_j and f_i as follows.

1) Shadowing error: A filter f_j has a shadowing error caused by f_i when

a) TR (f_i, f_j) = equal, f_i·action ≠ f_j·action,

b) TR (f_i, f_j) = inside, f_i·action ≠ f_j·action.

In this case, the filter f_j is never executed, as f_i prevents f_j from matching all the packets.

2) Redundancy error: A filter f_j has a redundancy error caused by f_i when

a) TR (f_i, f_j) = equal, f_i·action = f_j·action,

b) TR (f_i, f_j) = inside, f_i·action = f_j·action.

In this case, the filter f_j is never executed, as the filter f_i prevents f_j from matching all the packets that f_j can match. The redundant filters in the FP unnecessarily increase the size of the FP, and even removing f_j does not change the semantics of FP.

3) Correlation warning: A filter f_j has a correlation warning caused by f_i when

TR (f_i, f_j) = overlap, f_i·action ≠ f_j·action.

If filter f_j has a correlation warning caused by f_i, f_j is sometimes not executed for the packets that match the filter f_i.

4) Generalization warning: A filter f_j has a generalization warning caused by f_i when

TR (f_i, f_j) = contains, f_i·action ≠ f_i·action.

The filter f_j is executed only when packets arrive that does not satisfy f_i.

5) Redundancy warning: A filter f_j has a redundancy warning caused by f_i when

a) TR (f_i, f_j) = contains, f_i·action = f_j·action,

b) TR (f_i, f_j) = overlap, f_i·action = f_j·action

The filter f_j is sometimes executed, as f_i prevents f_j from matching some packets that can match f_j. However, if a filter f_j is disjoint to the other filters, then it does not fall within any of the above conflict classifications. In this case, we refer to it as a neither error nor warning filter.

6. BISCAL

BISCAL operates on the filter sets to extract the topological relationship of the filters. It treats the filter sets in a bit-vector format and uses seven primitive operators to find the topological relationship of the filters and three special vectors called characterization vectors to preserve the intermediate results. The main advantage of BISCAL is that it finds the disjoint filters in the intermediate stage of computation and removes those filters from the conflict computation. This is because a filter that is disjoint in any dimension is always disjoint in *n* dimensions.

6.1. Data Structure: Bit-Vector

An FP which consists of *m* filters is represented by a bit-vector $[b_0 \cdots b_{m-1}]$, where a bit b_i in the bit-vector represents the filter f_i. If a filter is selected from the FP, then the value of the corresponding bit is 1, and if not, then the value of the bit is 0. For example, let FP0 consist of $\{f_0, f_1, f_2, f_3, f_4\}$. If filters $\{f_1, f_3, f_4\}$ are selected from FP0, then the bit-vector is [01011]. Because this paper focuses on bit-vectors, hereinafter we will refer to a bit-vector as simply vector. The main reason for choosing the bit-vector representation is that it makes easier to apply logical operations to find the topological relationships between the filters. We introduce a function V2S that transforms the 1 s in the vector to its corresponding filters. For example, V2S ([11100]) = {f_0, f_1, f_2}.

6.2. Primitive Operators

There are seven primitive operators in BISCAL that compute the topological relationships of the filters. They are explained with example vectors v1 and v2, where v1= [1010] and v2 = [1011].

1) AND Operator (AND): This operator computes a bit-wise AND for a set of vectors. For example, **AND** ({v1, v2}) = **AND** ({[1010], [1011]}) = [1010].

2) Cartesian-AND Operator (C-AND): It computes the Cartesian product of two sets of vectors A and B and then computes the logical AND for the resulting vectors. **C-AND** (A, B) = {**AND** ((a, b)), | (a, b) ∈ A × B}.

3) OR Operator (OR): This operator computes a bit-wise OR for a set of vectors. For example, **OR** (v1, v2) = **OR** ({[1010], [1011]}) = [1011].

4) Counting one Operator (C1): This operator counts the number of 1s in an input vector. For example, **C1** (v1) = **C1** ([1010]) = 2.

5) NOT Operator (NOT): This operator returns the complement of an input vector. For example, **NOT** (v1) = **NOT** ([1010]) = [0101].

6) Pair-filters Operator (PF): This operator returns a set from two filter sets that contain the possible combinations of the two filters in each of the input vectors. For example, **PF** ([1010], [1011]) = {{f_0, f_2}, {f_0, f_2}, {f_0, f_3}, {f_2 f_3}}.

7) Permutation Operator (PO): This operator returns a set from two filter sets in which each filter is a filter selected from each of two input filter sets. For example, **PO** ($\{f_0\}$, $\{f_1, f_2\}$) = $\{\{f_0, f_1\}, \{f_0, f_2\}\}$.

6.3. Characterization Vectors

Characterization vectors are the vectors that characterize the topological relationship of the filters. There are two types of characterization vectors: 1) vectors characterizeing the topological relationship of the filters in the packet space, called CV and, 2) vectors characterizing the topological relationship of the filters projected on each axis of the packet space, called PCV.

6.3.1. Characterization Vectors for Filters in the Packet Space

CVs characterize the topological relationship of the filters in *n*-dimensional packet space. They consist of the following three kinds of vectors.

1) Co-Existence Vectors Set (OVS): OV is a vector in which a bit of the vector is 1 if the corresponding filter co-exists with another filter in some subspace of the packet space. OVS is a set of all OVs. For example, in **Figure 1**, f_0, f_1 and f_2 co-exist in S_4, therefore the vector representation is, OV = [11100].

2) Fully Covered Vector (FV): FV is vector in which the value of b_i is 1, if f_i is fully covered in *n*-dimensional space. For example, in **Figure** 1, f_0, f_1 and f_3 are fully covered in two-dimension and therefore, FV = [11010].

3) Disjoint Vector (DV): If a filter f_i is disjoint from all other filters in *n*-dimensional space, then the value of bit b_i is set to 1 in DV. For example, in **Figure 1**, f_4 is a disjoint filter and therefore DV = [00001].

6.3.2. Characterization Vectors for Projections of Filters on Each Axis of the Packet Space

PCVs are computed by using the projection of the filters on each axis of the packet space. The $X_i\{f_0, \cdots, f_{m-1}\}$ is an ordered set of the i^{th} predicates of the filters in FP and the projection of the filters on the i^{th} axis of the packet space corresponding to the i^{th} key, X_i. All the projected filters except for the default filter are decomposed in the boundaries specified by their i^{th} predicate. As a result of the decomposition, the axis is divided into multiple intervals. **Figure 3** shows an example of the spatial division for the projection of the filters in FP1 on the 0^{th} axis corresponding to the 0^{th} key X_0 or SrcIP and shows that six intervals, $I_0, \cdots, I_5$, are made by the decomposition of $X_0\{f_0, \cdots, f_5\}$.

Each interval has a set of filters, called sub-domain filter set, in which i^{th} predicate of the filter is always true within the interval. The sub-domain filter sets for all the intervals on the i^{th} axis except the empty sets form a set, named SFS_i. Each sub-domain filter set is transformed into a vector, and as a result, the SFS_i is transformed into a set of vectors, SVS_i. If a filter f_i exists in a sub-domain filter set, the corresponding bit of the vector b_i is set to 1. For example, $SFS_0 = \{\{f_0, f_1\}, \{f_0, f_1, f_2\}, \{f_2\}, \{f_2, f_3\}, \{f_4\}\}$ and $SVS_0 = \{[11000], [11100], [00100], [00110], [00001]\}$ for the spatial division for the projection of the filters in FP1 mentioned in the above.

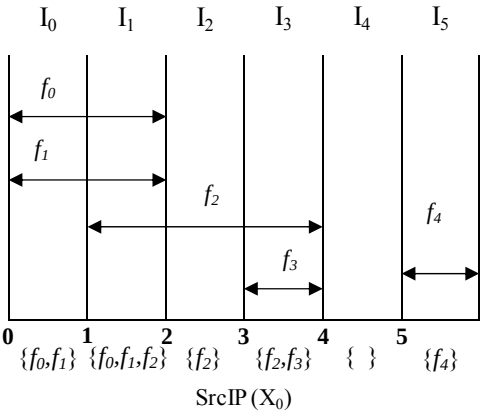


Figure 3. Spatial division for the projection of the filters in FP1 on the 0^{th} key X_0 or SrcIP.

PCV_i characterizes the topological relationship of the projected filters on the i^{th} axis and consists of the following three kinds of vectors.

1) $POVS_i$: POV_i is a vector in which a bit of the vector is 1 if the corresponding projected filter co-exists with another projected filter in some interval on the i^{th} axis of the packet space. $POVS_i$ is a set of all $POVS_i$s. For example, in **Figure 3**, $POVS_0$ = {[11000], [11100], [00110]}.

2) PFV_i: PFV_i is vector in which the value of bit b_i is 1 if the projection of f_i is fully covered on i^{th} axis of the packet space. For example, in **Figure 3**, the projections of $\{f_0, f_1\}$ and $\{f_3\}$ are fully covered and therefore, PFV_0 = [11010].

3) PDV_i: If a filter f_i is disjoint from all other filters projected on i^{th} axis of the packet space, then the value of b_i is set to 1 in PDV_i. For example, in **Figure 3**, f_4 is disjoint and therefore PDV_0 = [00001].

7. Implementation of Topology-Based Conflict Detection System

7.1. System Overview

The proposed system detects and classifies the conflicts in the given firewall policy, FP, which consists of *m* filters and *n* key fields. The default filter is the least priority filter f_{m-1} and is not considered for conflict detection

as it always conflicts with the remaining filters. Our system computes the topological relation of each filter pair (f_i, f_j), where f_j is the conflicting filter and f_i is the conflict causing filter. It then decides the errors and warnings for f_j according to the topological relation based on the conflict classification described in Section 4.

The overview of the proposed system is shown in **Figure 4** and consists of a vertical decomposer, spatial divisors, PCV extractors, a CV extractor, and a TR extractor.

The system receives the FP, and follows the procedure given below.

STEP 1: The vertical decomposer divides the FP into n sequences. Each sequence includes the i^{th} predicate of the filters in FP and represents projections of the filters in FP on the i^{th} axis of the packet space, $X_i(f_0,\cdots,f_{m-1})$, where i is from 0 to $n-1$.

STEP 2: The spatial divisor makes SFS_i by the spatial division of the projection $X_i(f_0,\cdots,f_{m-1})$ on the i^{th} axis in the way as described in the Section 6.3.2 for each projection where i is from 0 to $n-1$.

STEP 3: The PCV_i extractor transforms SFS_i into SVS_i in the way as described in the Section 6.3.2 and calculates PCV_i by applying the BISCAL to the vectors in the SVS_i where i is from 0 to $n-1$.

STEP 4: The CV extractor calculate the CVs by combining the results of PCVs with the BISCAL.

STEP 5: The TR extractor calculates the topological relationship among all combinations of two filters in the FP using BISCAL and classifies them into two types of errors, three types of warnings and others (neither errors nor warnings), as explained in Section 5.

The computation of the last three steps is taken over by BISCAL. It extracts the topological relationships for all the combinations of two filters from the CVs, and classifies them into five types of conflicts and others (neither error nor warning filters). The last three steps are explained in detail in the following subsections.

7.2. PCV Extractor

The PCV extractors calculate POVSs, PDVs and PFVs by the following steps.

STEP 3-1: $POVS_i$ is computed as follows:

Initialize $POVS_i = \{\emptyset\}$;

For all v $\in$ SVS_i, if **C1** (v) > 1, $POVS_i = POVS_i \cup$ v;

For example, $POVS_0 = \{[11000],[11100],[00110]\}$ for the SVS_0 calculated above.

STEP 3-2: PDV_i is calculated as follows:

PDV_i = **NOT** (**OR** ($POVS_i$));

For example, when $POVS_0$ is the value calculated above, PDV_0 = **NOT** $([11110]) = [00001]$.

STEP 3-3: In the calculation of PFV_i, the non-fully covered filters on the i^{th} axis of the packet space are computed initially and lastly the fully covered filters are computed. The non-fully covered filters are identified in NF_i by finding a vector in SVS_i with a single 1, because the non-fully covered filters are somehow alone in any subspace. For example, in **Figure 3**, f_2 is a non-fully covered filter, because f_2 is alone in I_2. In this way, the corresponding bits of non-fully covered filters are made 0, and the fully covered vector PFV_i is derived.

Initialize $NF_i = \{\emptyset\}$;

For all v $\in$ SVS_i, if **C1** (v) =1, $NF_i = NF_i \cup$ v;

PFV_i = **NOT** (**OR** (NF_i));

For example, in X_0, SVS_0 = {[11000], [11100], [00100], [00110], [00001]}, as mentioned above, NF_0 = {[00100], [00001]}, and PFV_0 = **NOT** ([00101]) = [11010].

7.3. CV Extractor

CV Extractor calculates the CVs using the PCV_i $(i=0,\cdots,n-1)$, calculated in the previous subsection. Before the computation of the CVs, we remove the already computed disjoint filters from $POVS_i(i=0,\cdots,n-1)$ and $PFV_i(i=0,\cdots,n-1)$. Using this technique, we improve our system efficiency, as unnecessary computations are removed by discarding

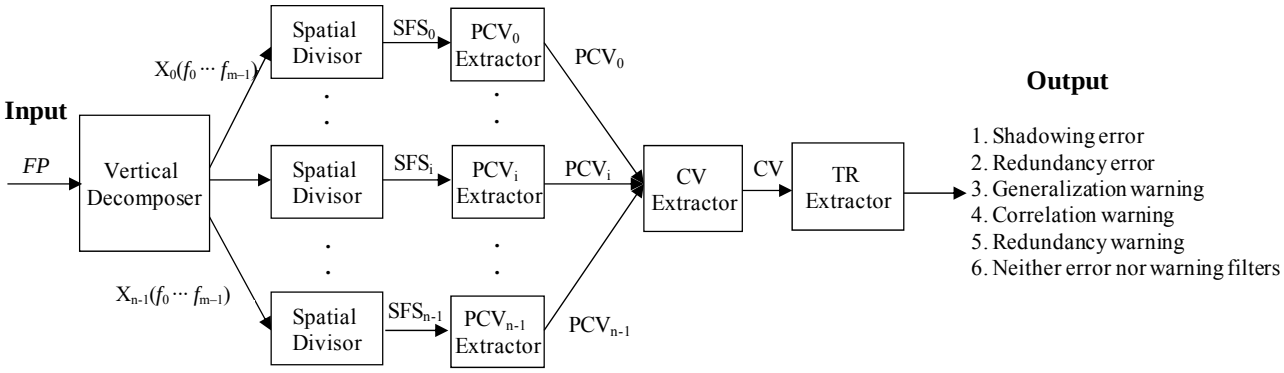


Figure 4. System overview of topology-based conflict detection system.

possibilities to make a filter pair with disjoint filters during the computation.

We introduce a vector DV', in which the value of a bit of the vector is 1 if the corresponding filter is disjoint in any one dimension. It is similar to DV because a bit in DV' is 1 if the corresponding filter is disjoint on the *n*-dimensional space, but is different from DV in that a bit in DV' might be 0 if the corresponding filter is disjoint on the *n*-dimensional space. The $POVS_i$ and PFV_i calculated in the previous subsection are replaced with OVS_i' and FV', respectively. Here, a bit becomes "0" if the corresponding filter is shown to be a disjoint filter in DV', in order to improve the efficiency of the computation.

STEP 4-1: DV', OVS_i' and FV' are calculated as follows. In this step, the disjoint filters are removed from OVS_i' and FV' to discard the possibility of a disjoint filter in the upcoming conflict detection steps.

$$DV' = \mathbf{OR}\left(PDV_0 \cdots PDV_i \cdots PDV_{n-1}\right) ;$$

For each i = 0 to $n-1$, OVS_i' = **C-AND** (**NOT** (DV'), $POVS_i$), and FV' = **AND** (**NOT** (DV'), PFV_i);

STEP 4-2: The OVS is calculated according to the following sub-steps, which calculate the intermediate results, IR, and then finally, the OVS.

$$OVS = \{\varnothing\} ;$$

$$IR_1 = \mathbf{C\text{-}AND}\left(OVS_0', OVS_1'\right);$$

For $i = 2$ to $n-1$, repeat

$$IR_i = \mathbf{C\text{-}AND}\left(OVS_i, OVS_{i-1}\right);$$

For all $v \in IR_{n-1}$, if $\mathbf{C1}(v) = 1$, $OVS = OVS \cup v$;

STEP 4-3: A filter is fully covered in an *n* dimension only when it is fully covered in all one dimensions, and therefore FV is computed as follows:

$$NF_n = \{\varnothing\} ;$$

For all $v \in IR_{n-1}$, if $\mathbf{C1}(v) = 1$, $NF_n = NF \cup v$;

$$FV = \mathbf{AND}\left(FV_0' \cdots FV_{n-1}', \left(\mathbf{NOT}\left(\mathbf{OR}\left(NF_n\right)\right)\right)\right);$$

STEP 4-4: DV is calculated in the same way as the vectors of one-dimension.

$$DV = \mathbf{NOT}\left(\mathbf{OR}\left(OVS\right)\right);$$

For example, the CVs for FP1, shown in **Table 1**, are computed for two dimension (SrcIP, DesIP), as follows: $OVS = \{[11000],[11100],[00110]\}$, FV = [11010] and DV = [00001].

7.4. TR Extractor

The TR Extractor calculates the sets of filter pairs, CCTP and $CCTP_{topology}$, so that we can obtain the topological relationships for all the filter pairs in the firewall policy from the above sets, and apply the rules for conflict classification defined in Section 5. Filter pairs that have the conflict causing topology are defined in CCTP as follows:

$$CCTP = \left\{\left(f_i, f_j\right) \middle| TR\left(f_i, f_j\right) \in \{\text{inside,contains,equal, overlap}\}\right\}.$$

Two filter pairs that have the same topological relationship are defined in $CCTP_{topology}$ in more detail, as follows:

$$CCTP_{topology} = \left\{\left(f_i, f_j\right) \middle| TR\left(f_i, f_j\right) = \text{topology}\right\},$$

where "topology" is one of inside, contains, equal, or overlap.

For example:

$$CCTP_{overlap} = \left\{\left(f_i, f_j\right) \middle| TR\left(f_i, f_j\right) = \text{overlap}\right\}.$$

Step 5-1: Computation of disjoint filters

The disjoint filters are computed in the DF as follows:

DF = **V2S** (DV);

For example, for FP1, DF = **V2S** ([00001]) = $\{f_4\}$, which shows that f_4 is a disjoint filter of FP1.

Step 5-2: Computation of CCTP

$$CCTP = \{\varnothing\} ;$$

For each v $\in$ OVS, CCTP = CCTP $\cup$ **PF** (v);

For example, the CCTP of FP1 is as follows:

$$OVS = \{[11000],[11100],[00110]\},$$

$$CCTP = \{(f_0, f_1),(f_0, f_2),(f_1, f_2),(f_2, f_3)\}.$$

STEP 5-3: Classification of topology between filter pairs

Initially, we calculate the fully covered filter pairs, and then lastly, each $CCTP_{topology}$.

STEP 5-3-1: Computation of fully covered filter pairs

We introduce two sets, S and SP, where S is a set of all fully covered filters in FP, and SP is a set of fully covered filter pairs. If f_j is a fully covered filter in S, and f_i ($i < j$) is some filter in FP, a filter pair (f_i, f_j) is in either one of $CCTP_{inside}$, $CCTP_{contains}$, or $CCTP_{equal}$. The computational steps for S and SP are as follows:

Initialize $SP = \{\varnothing\}$;

S = **V2S** (FV);

for each f_k of S, do

{$FF_k = [11\cdots1]$;

for each v $\in$ OVS,

if b_k of v is 1, FF_k = **AND** (FF_k, v);

Set b_k to 0 in FF_k;
SP = SP ∪ **PO** ({f_k}, **V2S** (FF_k));}

For example, the SP for FP1 is calculated as follows: S= **V2S** ([11010]) = {f_0, f_1, f_3} and OVS= {[11000], [11100], [00110]}. Then, FF_0 = **AND** {[11111], [11000], [11100]} = [11000].

Similarly, FF_1 and FF_3 are calculated as follows: FF_1 = **AND** ([11111], [11000], [11100]) = [11000], FF_3 = **AND** ([11111], [00110]) = [00110].

Set f_k bit to 0 in FF_k, and therefore, FF_0 = [01000], FF_1 = [10000] and FF_3 = [00100].

$$\mathrm{SP} = \mathbf{PO}\left(\{f_0\},\{f_1\}\right) \cup \mathbf{PO}\left(\{f_1\},\{f_0\}\right) \cup \mathbf{PO}\left(\{f_3\},\{f_2\}\right) = \left\{\{f_0, f_1\},\{f_3, f_2\}\right\}.$$

STEP5-3-2: Computation of $CCTP_{topology}$

All the combinations of filter pairs are classified using the values calculated above, according to their topological relationships.

$$\mathrm{CCTP}_{\mathrm{equal}} = \left\{ (f_i, f_j) \middle| f_i, f_j \in S, (f_i, f_j) \in \mathrm{SP} \right\};$$

$$\mathrm{CCTP}_{\mathrm{inside}} = \left\{ (f_i, f_j) \middle| f_i \notin S, f_j \in S, (f_i, f_j) \in \mathrm{SP} \right\};$$

$$\mathrm{CCTP}_{\mathrm{contain}} = \left\{ (f_i, f_j) \middle| f_i \in S, f_j \notin S, (f_i, f_j) \in \mathrm{SP} \right\};$$

The $CCTP_{overlap}$ is calculated as follows: $CCTP_{overlap} = CCTP \cap SP$;

For example, the SP for FP1 calculated in the STEP 5-3-1 leads us to the following two sets:

$$\mathrm{CCTP}_{\mathrm{equal}} = \{(f_0, f_1)\}$$

and

$$\mathrm{CCTP}_{\mathrm{inside}} = \{(f_2, f_3)\}$$

and

$$\mathrm{CCTP}_{\mathrm{overlap}} = \{(f_0, f_2), (f_1, f_2)\}.$$

Step 5-4: Conflict Classification

By using the sets of filter pairs calculated in the previous steps, we can determine the topological relationships TR (f_i, f_j) between any pair of filters f_i and f_j, and according to the rules described in Section 5, any filter f_j is classified.

For example, the filters in FP1 are classified as follows: f_1 has a shadowing error caused by f_0, f_2 has a redundancy warning caused by f_0, and a correlation warning caused by f_1, f_3 has a shadowing error caused by f_2 and f_0 and f_4 have neither errors nor warnings.

8. Evaluation

We have evaluated our proposed system using a mathematical analysis with a special case. We have developed a prototype of our system in JAVA programming language and then performed an experimental analysis to evaluate the efficiency of the proposed system.

8.1. Mathematical Analysis

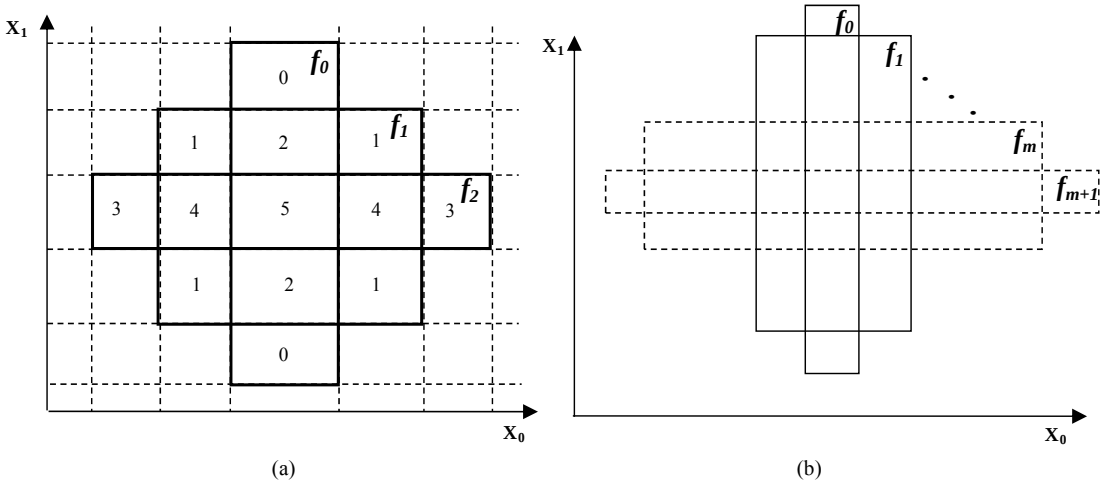
As we have discussed earlier, the efficiency of the conflict detection system is decided by the number of subspaces required to find the relationship between the filters. The basic difference in geometry and topology is that topology considers only the unique subspaces, whereas geometry considers all the subspaces. Therefore, the computation time and memory requirements for a topology-based approach are less than that of a geometry-based approach. But in general, it is difficult to mathematically analyze the difference between the computation time and memory requirements in geometry and topology-based approaches. Therefore, we selected an example policy, FPA, which includes a lot of conflicts, with each and every filter is symmetrical to each other, and every filter conflicting with all the other filters. The two-dimensional spatial representation of FPA with 3 filters is shown in **Figure 5(a)** and m + 1 filter is shown in **Figure 5(b)**.

In **Figure 5(a)**, for sake of simplicity, we have represented each subspace with a numerical digit. The subspaces with the same filter sets (non-unique subspaces) are represented by a single numerical digit. For example, the number zero refers the subspace of filter f_0, and the number four represents the subspace with filters {f_1, f_2}. Firstly, we compare the difference between the number of subspaces in **Figure 5(a)**. The computation of conflicts through topological approach requires only six subspaces, whereas the geometrical approach requires all thirteen subspaces for conflict detection. Likewise, if new filters are added by preserving the symmetrical structure, as shown in **Figure 5(b)**, we derived the difference in subspaces required for the topology and geometry approaches for **Figure 5**.

When an $m + 1^{th}$ filter is added in FPA, the number of new subspaces increases by four times the value of m in total and there exists m + 1 number of unique subspaces following the sum of natural number series. Therefore in topological approach, when the $m + 1^{th}$ filter is added, the number of subspaces considered for conflict detection is m + 1. However, in the geometrical approach, when the $m + 1^{th}$ filter is added, the total number of subspaces considered is four times the value of m. We can derive the following equations for the number of subspaces NS_i for both approaches.

Topology:

$$\mathrm{NS}_i(m) = \left(m^2 + m\right)/2$$



(a) (b)

Figure 5. FPA in two dimensions. (a) 3 filters; (b) *m* + 1 filters.

and

$$\mathrm{NS_i}(m+1) = \mathrm{NS_i}(m) + (m+1).$$

Geometry:

$$\mathrm{NS_i}(m) = m^2 + (m-1)^2$$

and

$$\mathrm{NS_i}(m+1) = \mathrm{NS_i}(m) + 4(m).$$

We have substituted different values of *m*, and tabulated the number of subspaces in **Table 2**. Our mathematical analysis shows that the number of subspaces for the topology approach is nearly one-fourth of the geometrical approach in two-dimensional space. The difference between the topology approach and the geometry approach is much larger when the dimension increases. In practical firewall policies, the efficiency of the topology approach is extremely high, because BISCAL removes the disjoint filters in the intermediate computation itself. We verify these using experiments in the next section.

8.2. Experimental Analysis

We have evaluated our system by performing a comparative analysis between the proposed system and conventional geometry-based approaches [7-9].

Table 2. Results of mathematical analysis.

Number of filters	Geometry	Topology
2	5	3
5	41	15
100	19801	5050
1000	1,998,001	500,500

Our experiments were performed on Intel (R) Core (TM) i5 CPU 750 @ 2.67 GHz 2.67 GHz with 4.00GM RAM running on Windows 7 professional. We conducted experiments with two policies, FPA and FPB. FPA is the policy discussed in the previous subsection, and FPB is a synthetic firewall policy, which is generated by adding a large number of filters to a small practical firewall. We have developed FPB based on the practical firewall policy being used in our lab. It consists of 99 packet filters of 5 dimensions, with 32-bit SrcIP, DesIP addresses, 16-bit SrcPort, DesPort numbers, and an 8-bit protocol. The synthetic firewall policy (FPB) ranges in size from 100 to 1000. In this paper, like other firewall management techniques [3-22], we did not consider the stateful filters for experimental evaluation. The treatment of conflict detection in stateful firewalls is a topic for future work. We conducted three experiments with both FPA and FPB.

Exp.1: Comparative performance analysis of topology and geometry [7-9] using FPA.

Exp.2: Evaluation of the system behavior in different scenarios by varying the ratio of wildcards in FPB.

Exp.3: Evaluation of system behavior in practical firewall policies by varying the number of filters in FPB.

In the three experiments, we have measured parameters such as memory and computation time. Memory is compared by examining the number of subspaces (NS) required for conflict computation. Computation time is the measure of the program execution time until conflict classification. We conducted three experiments, as shown above, and plotted graphs showing the number of filters on the x-axis, and the computation time expressed in seconds and memory expressed in KB and MB on the y-axis. The results of Exp.1 are shown in **Figure 6**. It is clear from the graph that our proposed topology-based

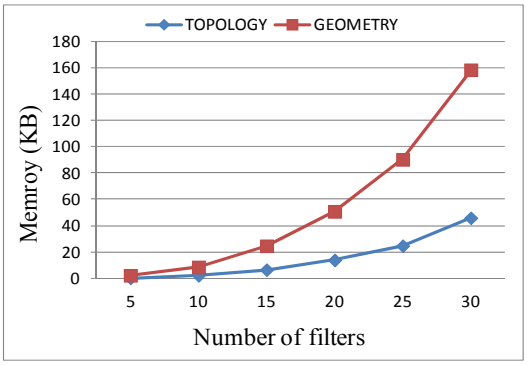


Figure 6. Comparison of topology and geometry.

system performs better than the geometrical approach.

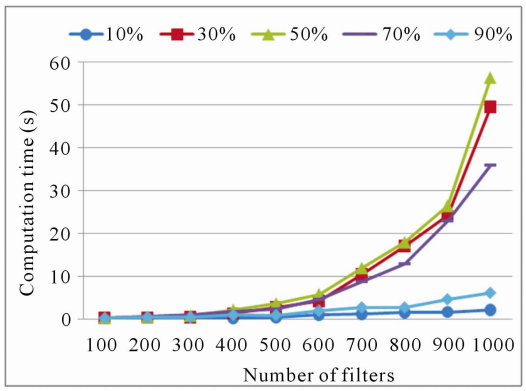
Exp.2 is performed by varying the ratio of the wildcards of the input filters, as shown in **Figures 7(a)** and **(b)**. This analysis is performed to examine how the system behaves with different kinds of polices used in various environments. We have synthesized various FPs by varying the distribution of wildcards in FPB.

We found that the computation time is less when the ratio of wildcards is in two extremes. When the ratio of wildcards is high, most of the filters occupy the *n*-dimensional space. As a result, there are only a few unique subspaces for conflict detection, as most of the subspaces have the same set of filters. When the ratio of wildcards is too low, most of the filters become disjoint to the others, and therefore the number of conflicting subspaces is less. Therefore the memory and computation time is less for lower and higher percentages of wildcards.

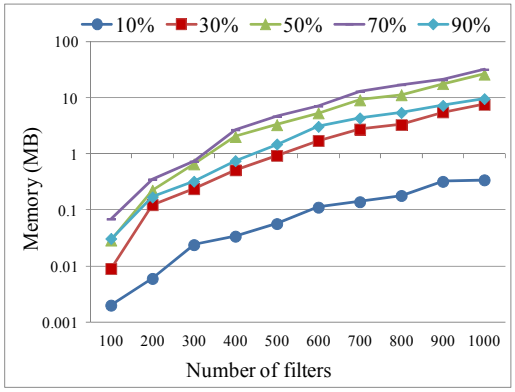
Exp.3 is performed using FPB to examine the system behavior with practical firewall policies. When the number of filters is increased, the system requires a reasonable computation time and memory when detecting conflicts, as shown in **Figures 8(a)** and **(b)**. For example, the system takes only 100 seconds to detect and classify the conflicts for $m = 500$.

9. Conclusions and Future Work

Our proposed topology-based conflict detection system detects the conflicts in FP and classifies them efficiently. It performs well as compared to conventional geometri-

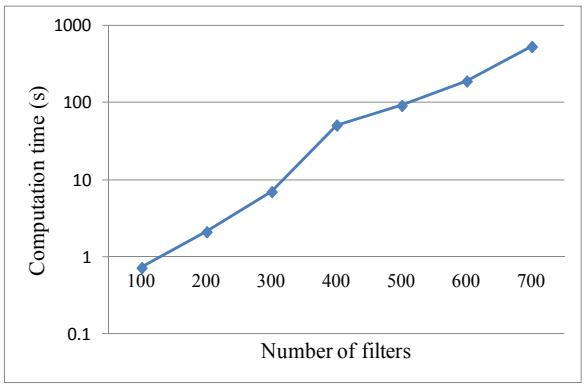


(a)

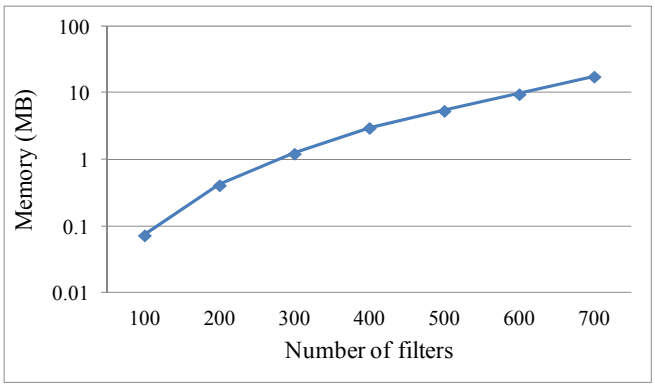


(b)

Figure 7. Performance analysis by changing the ratio of wildcard in FPB. (a) Computation time; (b) Memory.



(a)



(b)

Figure 8. Performance analysis of system using FPB. (a) Computation time; (b) Memory.

cal approaches and efficiently utilizes the two important resources: computation time and memory. Our future research plan focuses on the detection of conflicts caused by combinations of filters in firewall policies and conflict detection in stateful firewalls [28].

10. Acknowledgements

This research was partially supported by JSPS KAKENHI 23500085 and by the Hori Science and Art Foundation.

11. References

[1] The FreeBSD Documentation Project, Ipfw, http://freebsd.org/doc/enUS.ISO88591/books/handbook/firewalls-ipfw.html

[2] A. Wool, "A Quantitative Study of firewall Configuration Errors," *Computer*, Vol. 37, No. 6, 2004, pp. 62-67. doi:10.1109/MC.2004.2

[3] H. Hamed and A. I. Shaer, "Taxonomy of Conflicts in Network Security Policies," *IEEE Communications Magazine*, Vol. 44, No. 3, 2006, pp. 134-141. doi:10.1109/MCOM.2006.1607877

[4] A. Mayer, A. Wool and E. Ziskind, "FANG: A Firewall Analysis Engine," 2000 *IEEE Symposium on Security and Privacy*, Oakland, 14-17 May 2000, pp. 177-187.

[5] A. Wool, "Architecting the Lumeta Firewall Analyzer," *Proceedings of the* 10*th conference on USENIX Security Symposium*, Berkeley, August 2001, pp. 85-97.

[6] M. Yoon, S. Chen and Z. Zhang, "Minimizing the Maximum Firewall Rule Set in a Network with Multiple Firewalls," *IEEE Transactions on Computers*, Vol. 59, No. 2, 2010, pp. 218-230. doi:10.1109/TC.2009.172

[7] D. Eppstein and S. Muthukrishnan, "Internet Packet Filter Management and Rectangle Geometry," *Proceedings of the* 12*th Annual ACM-SIAM Symposium on Discrete Algorithms*, Washington, 2001, pp. 827-835.

[8] Y. Yin, Y. Katayama and N. Takahashi, "Detection of Conflicts Caused by a Combinations of Filters Based on Spatial Relationships," *The Information Processing Society of Japan*, Vol. 49, 2008, pp. 3121-3135.

[9] Y. Yin, R. S. Bhuvaneswaran, Y. Katayama and N. Takahashi, "Implementation of Packet Filter Configurations Anomaly Detection System with SIERRA," *International Conference on Information and Communication Security*, Beijing, 2005, pp. 467-480.

[10] T. Subana, Y. Yin, Y. Tateiwa, Y. Katayama and N. Takahashi, "A Topological Approach to Detect Conflicts in Firewall Policies," 2009 *IEEE International Symposium on Parallel & Distributed Processing*, Rome, May 2009.

[11] T. Subana, Y. Yin, Y. Tateiwa, Y. Katayama and N. Takahashi, "BISCAL: A Bit-Vector Based Spatial Calculus for Analyzing the Mis-Configurations in Firewall Policies," *IEICE Technical Report*, Vol. 108, No. 409, 2009, pp. 101-106.

[12] E. Al-Shaer and H. Hamed, "Conflict Classification and Analysis of Distributed Firewall Policies," *IEEE Journal on Selected Areas in Communications*, Vol. 23, No. 10, 2005, pp. 2069-2084. doi:10.1109/JSAC.2005.854119

[13] V. Capretta, B. Stepien, A. Felty and S. Matwin, "Formal Correctness of Conflict Detection for Firewalls," *Proceedings of the* 2007 *ACM Workshop on Formal Methods in Security Engineering*, Virginia, November 2007, pp. 22-30.

[14] H. K. Verizon and K. A. Ahmat, "Fast and Scalable Method for Resolving Anomalies in Firewall Policies," 14*th IEEE Global Internet Symposium* 2011, Shanghai, 15 April 2011, pp. 839-844.

[15] H. Hu, G. J. Ahn and K. Kulkarni, "FAME: A Firewall Anomaly Management Environment," 3*rd ACM Workshop on Assurable and Usable Security Configuration*, Chicago, October 2010, pp. 17-26.

[16] A. X. Liu and M. G. Goudam, "Complete Redundancy Detection in Firewalls," *Proceeding of* 19*th Annual IFIP Conference on Data and Applications Security*, Storrs, 2005, pp. 196-209.

[17] K. Matsuda, "A Packet Filtering Rules Compression by Decomposing into Matrixes," in Japanese, *The Information Processing Society of Japan*, Vol. 48, No. 10, 2007, pp. 3357-3364.

[18] K. Golnabi, R. K. Min, L. Khan, and E. Al. Shaer, "Analysis of Firewall Policy Rules Using Data Mining Techniques," 10*th IEEE/IFIP Network Operations and Management Symposium*, Vancouver, 3-7 April 2006, pp. 305-315.

[19] L. Yuan, J. Mai, Z. Su, H. Chen and P. Mohapatra, "FIREMAN: A Toolkit for Firewall Modeling and Analysis," *Proceeding of* 2006 *IEEE Symposium on Security and Privacy*, Oakland, 21-24 May 2006, pp. 199-213.

[20] B. Zhang, E. AI. Shaer, R. Jagadeesan, J. Riely and C. Pitcher, "Specifications of a High-Level Conflict-Free Firewall Policy Language for Multi-Domain Networks," *Proceedings of the* 12*th ACM Symposium on Access control Models and Technologies*, Sophia Antipolis, 20- 22 June 2007, pp. 185-194.

[21] A. Hari, S. Suri and G. Parulkar, "Detecting and Resolving Packet Filter Conflicts," *Proceeding of* 19*th Annual Joint Conference of the IEEE Computer and Communications Societies*, Tel Aviv, 26-30 March 2000, pp. 1203-1212.

[22] F. Baboescu and G. Varghese, "Fast and Scalable Conflict Detection for Packet Classifiers," 10*th IEEE International Conference on Network Protocols*, Paris, 12-15 November 2002, pp. 270-279. doi:10.1109/ICNP.2002.1181414

[23] N. Takahashi, "A Systolic Sieve Array for Real-Time Packet Classification," *The Information Processing Society of Japan*, Vol. 42, No. 2, 2001, pp. 146-166.

[24] T. Srinivasan, N. Dhanasekar, M. Nivedita, R. Dhivyakrishnan and A. A. Azeezunnisa, "Scalable and Parallel Aggregated Bitvector Packet Classification Using Prefix Computation Model," *Proceedings of the international symposium on Parallel Computing in Electrical Engi-*

neering, Bialystok, 13-17 September 2006, pp. 139-144.

[25] T. V. Lakshman, "High-Speed Policy Based Packet Forwarding Using Efficient Multi-Dimensional Range Matching," *Proceedings of the ACM SIGCOMM*'98 *Conference on Applications*, *Technologies*, *Architectures*, *and Protocols for Computer Communication*, Vancouver, 2-4 September 1998, pp. 203-214.

[26] S. Singh, F. Baboescu, G. Varghese and J. Wang, "Packet Classification Using Multidimensional Cutting," *Proceedings of the* 2003 *Conference on Applications*, *Technologies*, *Architectures*, *and Protocols for Computer Communications*, Karsruhe, 7 February 2003, pp. 213-224.

[27] M. Buddhikot, S. Suri and M. Waldvogel, "Space Decomposition Techniques for Fast Layer-4 Switching," *Proceedings*: *Protocols for High-Speed Networks*, Salem, August 1999, pp. 25-42.

[28] K. Hida, Y. Katayama and N. Takahashi, "A Filter Reverse Search System for LANs with Stateful Firewalls," *IEICE Technical Report*, Vol. 107, No. 483, 2007, pp. 65-70.

[29] Max J. Egenhofer, "A Formal Definition of Binary Topological Relationships," 3*rd International Conference on Foundations of Data Organization and Algorithms*, Vol. 367, 1989, pp. 457-472.

