A Topological Approach to Implement a Conflict Detection System for Time-based Firewall Policies

時間ベースファイアウォールポリシのコンフリクト検出システムの実現へのトポロジカルアプローチ

by

SUBANA THANASEGARAN

A DISSERTATION
submitted to the Graduated School of Engineering,
Department of Computer Science and Engineering,
Nagoya Institute of Technology
in a partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
written under the supervision of
Professor NAOHISA TAKAHASHI

NAGOYA INSTITUTE OF TECHNOLOGY
Nagoya, Japan
November 2011

Abstract

Firewall policy management is a notoriously difficult task for any network administrator as it is always under modification. Recently, time-based filters are widely used to restrict the network traffic based on certain times of the day or certain days. The firewall policies with time-based filters are called as time-based firewall policy, TFP. Due to the introduction of TFP, the management becomes further complicated and as a result misconfigurations like conflicts appear when a packet matches multiple filters. It makes some filters redundant or shadowed and therefore the actual configuration is not reflected in the network. Even though conflict detection

for firewall policies is available based on spatial analysis through geometrical approach, when the number of filters and headers increases it takes huge computation time and memory. The other problem is the prevailing conflict detection techniques cannot deal with time-based filters, and as a result, when they are applied to time-based filters, the time domain is ignored; therefore, the problem of obtaining false positive results arises, stating a non-conflict as a conflict. This problem has not been addressed in any research regardless of its significance. To solve these problems, we have presented the following three analyses: (1) A Topology-based spatial analysis for conflict detection in FPs through a Bit-vector Based Spatial Calculus named BISCAL. (2) A Geometry-based Temporal Analysis. (3) A Topology-based Temporal Analysis. The first analysis solves the problem of high computation time and memory. Even though, the second and third analysis solves the problem of the ignorance of time field, it is necessary to combine the spatial and the temporal analyses for the conflict detection in TFPs. Therefore we have proposed two architectures as follows: (1) Simultaneous Analysis, and (2) Iterative Analysis. We have evaluated the feasibility and usefulness of the detection systems through experimental analysis and validated the performances of the systems.

The above three analyses and two architectures give a new methodology for detecting conflicts in TFPs. It gives a direction that makes it possible to implement TFP diagnosis tools based on the methodology in practice and brought a new era in the field of conflict detection in TFPs.

Acknowledgments

At first, I would like to take this opportunity to thank Japan Ministry of Education, Culture Sports, Science and Technology (MEXT) for awarding me the Japanese Government Monbukagakusho Scholarship for my Masters and Doctors program. I am much honored to be the recipient for this award. Receiving this scholarship motivates me to maintain a peaceful life and providing me confidence and willingness to help me achieve my goals.

I would like to record my gratitude to Professor Naohisa Takahashi for his supervision, advice, and guidance from the very early stage of this research as well as giving me extraordinary experiences throughout my research work. His truly scientist intuition has made him as a constant oasis of ideas and passions in science, which exceptionally inspire and enrich my growth as a student, a researcher and a scientist want to be. I am indebted to him more than he knows.

I would like to thank Professor Yoshiaki Katayama, for giving me suggestions in my seminar hours, correcting my papers and helping me with the research. Above all and the most needed, he provided me unflinching encouragement and support in various ways. I would like to thank Professor Yuichiro Tateiwa for his cooperation and support and providing me useful comments for my papers and helping me in Japanese Language. I also would like to extend my gratitude to Professor Koichi Wada for providing me valuable comments and suggestions during my presentations to improvise my research work.

I also thank the past and present members of Takahashi Laboratory who have directly or indirectly provided me with the appropriate environment for the completion of this work. I would like to thank my lab mates and many thanks go in particular to Mrs. Yi Yin who helps me whenever I ask any kind of doubts and any kind of help at any time and her crucial contribution, which made her a backbone of this research and so to this thesis. Her involvement with her originality has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come. I have enjoyed scientific discussions with Mr.

Pablo who sits next to me and I greatly value his friendship and extend my gratitude for his constant encouragement. I would like to thank my senior Mr. Karthik Jeyakodi who guided me about the lab environment and makes me to habituate with the Japan life at the initial stage. I would like to acknowledge and extend my gratitude to our lab secretary Junko Koike for taking care of all my official matters. At the foreign student's center of Nagoya Institute of Technology, I would like to express my gratitude to all the staff members for helping me with various aspects from daily life in Japan to learning the Japanese language itself. This helped eventually during discussions with my professors and lab mates about my research work. I would also like to thank especially Mrs. Imai and Mrs. Yamamoto for helping me in my entire study curriculum.

My special thanks go to my uncle Dr. Arulkumaran who introduced me to Nagoya Institute of Technology to pursue my higher studies. I would like to thank Mrs. Karthik Jegajothi for being introduced into my lab. It is a pleasure to express my gratitude wholeheartedly to Dr. Venkatesan's family, Dr. Mahesh's family for their kind hospitality during my stay in Japan. I would like to extend my special thanks to my near and dear friends like Mr. Madan and his wife Rekha, Ms. Ulka Suryavanshi and to my cousin Ms.Vijitha Thangaraja for their moral support towards my journey in Japan.

Finally, my family deserves special mention for their inseparable support and prayers. I would like to thank my husband Mr. Sridharbabu whose dedication, love and persistent confidence in me, has taken the load off my shoulder. I owe him for being unselfishly let his intelligence, passions, and ambitions collide with mine. I would like also to thank his parents and sister for their warmth and love towards me. I would like to sincerely thank my father Mr. Thanasegaran who put the fundament my learning character, showing me the joy of intellectual pursuit ever since I was a child and my mother Mrs. Leela Thanasegaran is the one who sincerely raised me with her caring and gentle mother's love. I would like to express my gratitude to my younger sister Ms. Suganya Thanasegaran for being so affectionate and loving and keep on encouraging me during my curriculum.

Finally, I would like to thank everybody who was important to the successful realization of thesis, as well as expressing my apology that I could not mention personally one by one.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Without a well managed firewall, a computer can be easily hijacked in few seconds. Firewalls have been able to address most network security issues and are used in organizations to protect the corporate intranet. Firewalls are used to filter traffic to and from the trusted enterprise network on the basis of the firewall policies defined keeping in mind the overall security policy of the organization.

Packet filtering technique in firewall provides initial level of security and operates on the network layer or the internet layer. In packet filtering network packets are accepted or denied based upon the predefined ordered set of filters, called a firewall policy FP. Each filter '*f*' in FP has a condition and an action (accepts or deny). When the key fields of the header of an incoming packet P satisfy the conditions of the filter *f*, the filter's action will be carried out on the incoming packet. A first matching scheme is used by many packet filtering systems such as IPFW in FreeBSD [33]. When the firewall finds a first matching filter *f* satisfies P's header fields, it applies the filter's action to the packet P.

Recently, Time-based filters are widely in use to control network traffic in time [34-36]. It is actively used in many applications like Access Control Lists, Surf control web filter and Linux iptables [36]. This is very handy when a service is required to be available only at certain times of a day or even certain days. For example, they can allow better QOS in the accounting department during the last three days of a month and can shutdown certain social networking sites during business hours. A firewall policy with time-based filters is called as a Time-based Firewall Policy TFP.

Although the deployment of firewall technology is the first important milestone toward securing networks, the effectiveness of firewall security may be limited or compromised by a poor management of firewall policy rules. One of the interesting problems is that how extent the rules are useful, up-to-dated, well-organized or efficient to reflect current characteristics and volume of network packets. For example, the network traffic trend may show that some rules are out-dated or not used recently. This may further lead one to consider removing, aggregating or reordering of the rules to optimize the firewall policy and efficiency. Also, server and network logs may validate or confirm that firewall policy rules are updated and consistent with the current network services.

It is a highly tedious job for the administrator to maintain and manage firewall policies as always it is subjected to modification. For example, in examining 37 firewalls in production enterprise networks, Wool found that all the firewalls were misconfigured and vulnerable [2]. Among many misconfigurations, conflict is a misconfiguration that often occurs in firewall polices. When a packet matches two or more filters in the firewall policy, conflict occurs. Due to the presence of conflicts in the firewall policies, the lower priority filter in the firewall policies will be never executed (**error**) and sometimes executed (**warning**). These kinds of conflicts create potential security holes and bring unintended traffic into the network as conflicting policies do not reflect the administrator's intention. Hazem *et al*. found that there is a high probability of creating conflicts even by expert system administrators and network practitioners [10]. *Therefore it is required to bridge the gap between what is written in the firewall policy rules and what is being observed in the network.*

1.1 Problem Definition

Various techniques have been developed to detect the conflicts in the firewall policies [4-26] to help the administrators. Hazem *et al* developed a conflict detection algorithm which computes the relationship between two filters based on the result of subsequent comparisons [10]. It detects conflicts by comparing the relationship between every two filters. Spatial analysis of firewall policies is also an interesting area of research for the problem of conflict detection. In this analysis, the *n*-key fields are projected in *n*-dimensional space and the *n*-dimensional topological relationships, TR^n of the filters are computed. The main advantage of spatial analysis is that the relationships between combinations of filters can be computed by projecting in the *n*-dimensional space at once other than the two filters and therefore a systematic and powerful conflict classification is possible. The two types of spatial analysis are geometrical analysis and topological analysis. Yin *et al* developed a conflict detection system based upon the spatial decomposition of the firewall policies [4]. The drawback of this approach is that when the number of filters and the matching key fields increases, it demands high memory usage and computation time. Therefore it is necessary to solve this problem by preserving the advantages of spatial analysis.

The other problem is that, the prevailing techniques cannot deal with the problem of conflict detection for TFPs. As a result, when they are applied to TFPs, the time domain is ignored; therefore, the problem of obtaining false positive results arises, stating a non-conflict as a conflict. This problem has not been addressed in previous researches regardless of its significance. When we summarize the above discussions, the two main problems are as follows: **(1) Large consumption of memory and computation time to detect conflicts through spatial analysis, (2) Presence of false positive results during conflict detection for TFPs.**

1.2 Research Aims and Approaches

To solve the above problems, our main research goals are: (1) To develop a conflict detection technique based upon spatial analysis with low computation time and memory, (2) To develop a conflict detection system for TFPs to discard the false positive results. To achieve our research goals, we have studied the importance of spatial analysis of firewall policies and investigated the possibility of the conflict detection through topology. We have found that the topology-based conflict detection can able to discard the drawbacks in the geometrical approach by preserving the advantages of spatial analysis. And also, we have made a thorough research on temporal analysis through geometry and topology and developed the following techniques to solve the above problems:

P1: A topology-based conflict detection through a Bit-vector based Spatial Calculus BISCAL.

P2: A geometry -based temporal analysis.

P3: A topology-based temporal analysis.

By our first proposal, we have solved the first problem which takes less computation time and memory than the previous geometry-based conflict detection technique. It computes the n-dimensional topological relationships, TR^n of the filters to detect conflicts. To practically implement the conflict detection system, we have proposed an implementation method called a bit-vector based spatial calculus named BISCAL. It is a systematic and simple way of extracting the TR^n of the filters through bit-vector based logical operations. To solve the second problem, it is necessary to analyze the time domain and therefore we have proposed the analyses P2 and P3.

The P2 and P3 analyze the TFPs in time domain through geometry and topology respectively. In the geometry-basis, the filters are projected in a predetermined conflict detection period and the filters are divided in their boundaries and the temporal analysis is performed. The disadvantage of this analysis is that when the conflict detection period is long, it takes large computation time and memory. To solve this problem, we have proposed a topology-based temporal analysis. In this analysis, there is a mapping mechanism to project the filters in time domain. It removes the unnecessary repetitions of the periodic time-based filters when the conflict detection period is long.

Even though we have proposed techniques to analyze TFPs in the time domain, it is necessary to combine the spatial and the temporal analysis to compute the conflicts in TFPs. To achieve the goal, we have proposed the following architectures in which the analysis in time and space can be combined: (1) Simultaneous Analysis, (2) Iterative Analysis. In both the architectures either of the two temporal analyses can be incorporated.

In the simultaneous analysis, the TFPs are analyzed in time and space domains at the same time and the TR^{n+1} of the filters are computed. In the Iterative Analysis, the time domain is divided into intervals by dividing the filter in their boundaries and for each time interval, the filter sets are analyzed in the *n*-dimensional space and finally the results of all the intervals are combined to find the TR^{n+1} of the filters. We have proposed the extension of BISCAL which can compute the TR^{n+1} of the filters for TFPs.

We have evaluated the architectures by performing comparative and experimental analysis of the two system architectures and validated the better performance systems.

With our experimental results, we have found that the workload of the network administrator can be decreased and an effective management of time-based firewall policies can be achieved by practicing our conflict detection systems.

1.3 Related Work

Over the past few years, researches on the analysis of firewalls have received much attention [2-32]. Hazem *et al* developed a conflict detection algorithm which computes the relationship between two filters based on the result of subsequent comparisons [10]. The condition of every two filters is compared to find whether it makes any conflict or not. Matsuda have proposed a packet filtering rules compression model by matrix decomposition. The matrixes are the smallest hyper cubes into which a hyper space including a filter set is decomposed at all the boundary derived from the range values of the filters. Mapping these matrixes to the filters enables to find unnecessary filters [30]. Yin *et al* developed a conflict detection system based upon geometry-based spatial analysis of filters [4]. The filters are projected in n-dimensional space where the co-ordinates of the filters are the boundary values of the condition of a filter. The filters are decomposed in their boundaries into subspaces and the geometrical location of the filters in the n-dimensional space is analyzed to find the conflicts. The advantage of conflict detection through spatial analysis is that conflicts caused by combinations of filters can be computed. But, the disadvantage of the previous system is that when the number of filters and headers increases, it takes huge computation time and memory to compute the conflicts.

The two different types of spatial analysis are geometry-basis and topology-basis. In our research, we have thoroughly studied the geometry-based and topology-based spatial analysis of filters in n-dimensional space. All the previous works mentioned above can be considered as a geometry-based analysis because they treat with the location of the filters, i.e., the range values specified in the condition of the filters during the analysis of conflict detection. In our research, we have found that analyzing the topology of the filters is sufficient enough to find the conflicts in the filters. For example, rather than analyzing the geometrical location of the filters, the essential subspaces that are necessary to extract the topology of the filters is sufficient to extract conflicts. By this way, we can able to reduce the number of subspaces that is required for conflict detection and also the advantage of finding the conflicts between the combinations of filters is also possible. Thereby, we discard the drawbacks of the geometry-based spatial analysis, and developed a topology-based spatial analysis, P1 for conflict detection in FPs. We have implemented it through a bit-vector based spatial calculus named BISCAL.

Even though, a lot of techniques are available to detect conflicts in FPs, there is no technique prevailing to find the conflicts in TFPs. If the TFPs are analyzed with available conflict detection techniques, false positive results arise saying a non-conflict as conflict. It occurs due to the ignorance of time domain during conflict detection. Therefore we have proposed a geometry-based temporal analysis to analyze the filters in the time domain in P2. In the geometry-based analysis, when the conflict detection period is long, the periodic time-based filters repeat ample number of times, and takes huge computation time and memory. To solve this problem, we have proposed a topology-based temporal analysis in P3. Our research contributions for the conflict detection of TFPs in the space and time domains are shown in Table 1.1.

To compute the conflicts in TFPs, it is necessary to analyze both the space and time domain. Therefore, we have proposed the following architectures to combine the analysis of topology-based spatial analysis and the temporal analysis: (1) Simultaneous Analysis, and (2) Iterative Analysis. In both the architectures, regarding the time domain, either one of P2 or P3 can be incorporated as shown in Table 1.2. By the above three analyses and two architectures, we could able to solve the problems discussed in the related work and have contributed a new era for the conflict detection in TFPs.

Table 1.1 Analyses in Conflict Detection

Domain / Spatial Analysis	Space	Time
Geometry	Yin et al [4], Eppstein et al [9]	P2
Topology	P1	P3

Table 1.2 Architecture Proposals

Architecture	Space	Time
Simultaneous Analysis	P1	P2 (or) P3
Iterative Analysis	P1	P2 (or) P3

1.4 Thesis Organization

This research produces several conferences and journal papers and this work is a compilation of those papers' propositions and results. The papers' material is rearranged in a more comprehensive order and a more extensive review of the literature was made.

Chapter 2 presents the conflict detection in the firewall policies, which introduces the internal form of a TFP, spatial decomposition of firewall policies, temporal analysis and the spatiotemporal analysis.

Chapter 3 presents the proposed system. It first introduces the overview of the proposed system and discusses the essential components in the proposed system in detail.

Chapter 4 describes the topology-based spatial analysis, and the implementation method through BISCAL and its evaluations.

In Chapter 5 discusses the topology-based temporal analysis and the geometry-based temporal analysis.

Chapter 6 discusses the spatiotemporal analysis through simultaneous and iterative analysis in detail and the evaluations of the two architectures.

Chapter 7 presents the conclusions of this dissertation and presents suggestions for future work.

Chapter 2

Conflict Detection in Time-based Firewall Policies

A firewall protects the network from unauthorized access and facilitates a secure access to the outside world. The packet filtering technique in a firewall provides a basic level of security and operates on the network layer of the OSI model or at the IP layer of the TCP/IP model. It controls the network traffic using a predefined, ordered set of filters called firewall policy, **FP**. Every filter *f* has a *condition* and an *action*. The condition consists of '*n*' predicates and the action can be either accept or deny. Each predicate in the condition is written based on the values contained in each '*n*' key field of a packet header. A packet *P* matches a filter *f* if and only if the packet header satisfies all the predicates in the condition of the filter.

2.1 A Time-Based Firewall Policy, TFP

Recently, time-based firewall policies have been introduced to control the network traffic in time. They are actively used in many applications like CISCO ACLs, surf control web filters, and Linux iptables. They are very handy when a service is required to be available only at certain times during the day or even certain days. For example, they can allow better QOS in the accounting department during the last three days of a month and also control certain unintended web services during business hours. A firewall policy with a time field is called time-based firewall policy, **TFP**. The time-field specifies the time constraint, which restricts the network traffic at specific dates and times. A packet *P* matches a time-based filter *f* if and only if the arrival time of *P* satisfies the time constraint of *f* and the values of *P*'s key fields satisfy all the predicates in the condition of *f*.

We have formally designed a TFP that consists of an ordered set of *'m'* filters and is expressed as follows: TFP: $(f_0, f_1, \ldots, f_{m-1})$. Consider the time-based filters f_i and f_j (i, j ⍰ $[0, m - 1]$, $i < j$), where the filter f_i is placed before f_j in the TFP. We adopt the first matching filtering scheme where the priority of the filter decreases from f_0 to f_{m-1}. Each filter f_i consists of a condition and an action. The condition consists of $(n + 1)$ predicates, $(p_{i0}, p_{i1},\ldots,p_{in})$, and f_i is expressed as follows:

$$f_i\text{: } p_{i0}, p_{i1},\ldots,p_{in}, \mathit{action},$$

where p_{i0} to p_{in-1} are the constraints for the values of the key fields to be used in packet filtering and p_{in} is a predicate that specifies a time constraint. The commonly used key fields are source IP addresses (represented as **SrcIP**), destination IP addresses (**DesIP**), source ports (**SrcPort**), destination ports (**DesPort**), and protocols (**Pro**).

Each predicate p_{ik} ($i \in [0, m-1]$, $k \in [0, n-1]$) can be represented as an exact value, a prefix, a range value, or a list in many firewall systems. However, in this thesis, we use only the range value for the sake of simplicity, because a filter with predicates in other forms can be easily converted into one or multiple filters with range values. Each predicate p_{ik} ($i \in [0, m-1]$, $k \in [0, n-1]$) 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 predicate p_{in} is the time constraint that the packet arrival time must satisfy to match the filter. It is represented as ***isActive***(f_i, T), which is defined below by using a time constraint; the time constraint which consists of a time range $[ts_i, te_i)$, a date d_j, and a set of days of the week S_i as well as $T = (t, de, dy)$, where t, de, and dy are the time, date, and day of the week when a packet arrives at the firewall.

[**isActive**(f_i,T)]

$$\textbf{isActive}(f_i,\mathrm{T}) = ((\mathrm{ts_i} \le \mathrm{t} < \mathrm{te_i}) \wedge (\mathrm{de} = \mathrm{d_i})) \vee ((\mathrm{ts_i} \le \mathrm{t} < \mathrm{te_i}) \wedge (\mathrm{dy} \in \mathrm{S_i})).$$

At a given time and date T, we say that a filter f_i is **active** if ***isActive***(f_i,T) is TRUE. We also say that a packet P satisfies the time constraint of a filter f if P arrives at the firewall when the filter f is **active**. Now, we define a predicate ***isMatched***(P, T, f_i), which shows whether a packet with arrival time and date T matches a filter f_i as follows.

[**isMatched**(P, T, f_i)]

$$\textbf{isMatched}(\mathrm{P}, \mathrm{T}, f_i) = (\mathrm{a_{i0}} \le \mathrm{u_0} < \mathrm{b_{i0}}) \wedge \ldots \wedge (\mathrm{a_{in-1}} \le \mathrm{u_{n-1}} < \mathrm{b_{in-1}}) \wedge \textbf{isActive}(f_i,\mathrm{T}).$$

We represent a filter in the form called **internal form**, which includes the range values instead of the predicates in (n + 1) fields**.** Since we can use a date and days of the week alternatively in the $(n + 1)^{th}$ field, i.e., the time field, there are two types of forms to represent a filter as follows:

(1) Type-1: **periodic** filter

f_i: $[a_{i0}, b_{i0}), [a_{i1}, b_{i1}),\ldots,[a_{in-1}, b_{in-1}), ([ts_i\ te_i), S_i)$, *action,*

(2) Type-2: **non-periodic** filter

f_i: $[a_{i0}, b_{i0}), [a_{i1}, b_{i1}),\ldots,[a_{in-1}, b_{in-1}), ([ts_i\ te_i), d_i)$, *action*

The field $([ts_i\ te_i), S_i)$ in a type-1 filter and $([ts_i\ te_i), d_i)$ in a type-2 filter are called **ActTIME,** in which the subfield $[ts_i\ te_i)$ is called **TIME** and the subfields S_i and d_i are called **DAY.** ts_i and te_i in the subfield TIME represent the start and stop values, respectively. The time and date are represented in a 24-h format hh:mm and DD/MM/YYYY, respectively. The days of the week are a subset of **S = {Sun, Mon,...,Sat}**. For example, S_i = {Mon, Fri}.

Figure 2.1 shows an example TFP written in the internal form, including the 0th field SrcIP, the 1st field DesIP, and the time field ActTIME. In this figure, the hyphen in the sub-field DAY shows **S**, which implies every day. We also represent the predicates and the action by ***f***f_i**.pname and** f_i**.action**, respectively, where pname is a predicate name**.** For example, f_3.ActTIME.TIME.start = 08:00, and f_1.action = Deny in the TFP shown in Figure 2.1.

In the example given in Figure 2.1, f_1 is **periodic** and becomes **active** from 12:00 to 14:00 on every Friday and f_2 is **periodic** and becomes **active** from 10:00 to 18:00 every day, whereas the default filter f_5 is always **active.** f_3 is **non-periodic** and becomes **active** from 8:00 to 12:00 on January 4, 2012. For example, when a packet P arrives at 08:00 on June 27, 2011, the four filters f_0, f_2, f_4, and f_5 become **active**. A packet matches a filter f from the above filters if the filter has the values u_0 and u_1 in SrcIP and DesIP fields such that $(f.a_{i0} \leq u_0 < f.b_{i0})$ ⍰ $(f.a_{i1} \leq u_1 < f.b_{i1})$. The default filter f_5 has the lowest priority, which denies access to all the packets when no other filter matches the packet.

	SrcIP	DesIP	ActTIME		Action
			TIME	DAY	
f_0	[0,5)	[0,5)	[08:00, 12:00)	Mon, Fri	Accept
f_1	[2,4)	[2,4)	[12:00, 14:00)	Fri	Deny
f_2	[2,4)	[2,4)	[10:00, 18:00)	-	Accept
f_3	[5,6)	[6,8)	[08:00, 12:00)	04/01/2012	Deny
f_4	[0,2)	[0, 2)	[10:00, 14:00)	-	Deny
f_5	$[0,2^{32})$	$[0, 2^{32})$	[00:00, 24:00)	-	Deny

Figure 2.1 Internal form representation of a sample TFP

2.2 Conflicts in TFP

A conflict occurs in a TFP when a packet matches multiple time-based filters. In our research we compute the *n*-dimensional topological relationships TR^n of the filters to find the conflicts in firewall policies. The n-dimension refers to the ‘n’ number of key fields in the firewall policies. As we know that the TFP has ‘n+1’ number of key fields, we also need to analyze the n+1th field – the time field to compute the conflicts in TFPs. Therefore we have to find the TR^{n+1} to find the conflicts in TFP. Initially, we explain TR^n of the filters and then we explain how TR^{n+1} is computed from TR^n. There are five topological relationships TR^n to identify conflicts between any pair of filters, f_i and f_j in FPs. In other words, TR^n (f_i, f_j) = {*disjoint, inside, contains, equal, overlap*}. The relations are shown in two dimensions in Figure 2.2, and can be generalized for higher dimensions. The filter space of a filter *f* is represented by FS (*f*).

Disjoint: TR^n (f_i, f_j) = disjoint when the intersection of the filters spaces is empty or FS (f_i) $\cap$ FS (f_j) =Ø, as shown in Figure 2.2(a).

Inside: TR^n (f_i, f_j) = inside when f_j is completely enclosed by f_i or FS (f_j) $\subset$ FS (f_i), as shown in Figure 2.2(b).

Contains: When f_i is enclosed by filter f_j, or FS (f_j) $\supset$ FS (f_i), then we say that there exists a relation TR^n (f_i, f_j) = contains between filters f_i and f_j, as shown in Figure 2.2(c). Contains is the converse of the inside relation.

Equal: TR^n (f_i, f_j) = equal when f_i and f_j are equal, or FS (f_i) = FS (f_i), as shown in Figure 2.2(d).

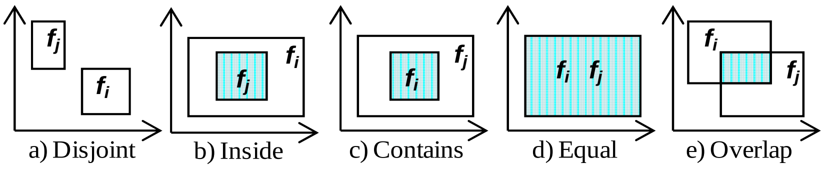


Figure 2.2 Topological relationships

Overlap: When f_i and f_j do not satisfy any one of the above four relationships, then we can say that TR^n (f_i, f_j) =overlap, as shown in Figure 2.2(e).

We introduce $\mathrm{TR}_{n+1}(f_i, f_j)$ as a topological relationship between filters f_i and f_j in the (n + 1)th dimension, the time field by using the temporal relationship between the time periods of the filters, in which isActive(f_i,T) = TRUE and isActive(f_j,T) = TRUE, as follows:

$\mathrm{TR}_{n+1}(f_i, f_j)$ =

Disjoint if **isActive**(f_i,T) = FALSE for all T such that **isActive**(f_j,T) = TRUE,

Equal if **isActive**(f_i,T) = TRUE for all T such that **isActive**(f_j,T) = TRUE, and **isActive**(f_j,T) = TRUE for all T such that **isActive**(f_i,T) = TRUE,

Inside if $\mathrm{TR}_{n+1}(f_i, f_j) \neq$ equal and **isActive**(f_i,T) = TRUE, for all T such that **isActive** (f_j,T) = TRUE,

Contains if $\mathrm{TR}_{n+1}(f_i, f_j) \neq$ equal and **isActive**(f_j, T) = TRUE for all T such that **isActive**(f_i,*T*) = TRUE,

Overlap otherwise.

A filter f_i has a single time period, in which ***isActive***(f_i,T) = TRUE, if it is non-periodic, i.e., it has a time and date in the time field, whereas it has infinite number of time periods if it is periodic, i.e., it has time and days of the week in the time field.

Now, we combine the *n*-dimensional topological relationships between the filters TR^n and the temporal relationship between the filters TR_{n+1} so that the topological relationship between two filters f_i and f_j in the $(n + 1)$-dimensional space TR^{n+1} can be determined as follows:

$$TR^{n+1}(f_i, f_j) = \begin{cases} \textbf{disjoint} & \text{if } TR^{n}(f_i, f_j) = \text{disjoint or } TR_{n+1}(f_i, f_j) = \text{disjoint}, \\ \textbf{equal} & \text{if } TR^{n}(f_i, f_j) = \text{equal and } TR_{n+1}(f_i, f_j) = \text{equal}, \\ \textbf{inside} & \text{if } TR^{n}(f_i, f_j) = \text{inside and } TR_{n+1}(f_i, f_j) = \text{inside}, \\ \textbf{contains} & \text{if } TR^{n}(f_i, f_j) = \text{contains and } TR_{n+1}(f_i, f_j) = \text{contains}, \\ \textbf{overlap} & \text{otherwise.} \end{cases}$$

The filter f_j is either partially or completely covered by f_i, except in the *disjoint* relation. In the *contains* and *equal* relations, f_j is completely covered by f_i and thus, f_i causes an **error** in f_j. In the *inside* and *overlap* relations, the filter f_i creates a **warning** for f_j as f_j is partially covered by f_i. Therefore, we can say that f_i and f_j do not cause any conflict when they have the *disjoint* relation.

Further, the conflicts are classified into two types of errors and three types of warnings according to $TR^{n+1}(f_i, f_j)$ and the action of the filters. We have been inspired by [10], which classifies conflicts into four types, and in addition, we have introduced a new type of conflict called **redundancy warning**. We have defined all the types of errors and warnings as follows:

(a) Shadowing error: There exists a shadowing error between f_i and f_j when any of the following conditions is true.

$$TR^{n+1}(f_i, f_j) = \text{equal, and } f_i.\text{action} \neq f_j.\text{action}$$

$$TR^{n+1}(f_i, f_j) = \text{contains, and } f_i.\text{action} \neq f_j.\text{action}$$

(b) Correlation warning: A filter f_i generates a correlation warning in f_j when the following condition is true.

`

$$TR^{n+1}(f_i, f_j) = \text{overlap, and } f_i.\text{action} \neq f_j.\text{action}$$

(c) Generalization warning: There exists a generalization warning in f_i and f_j when the following condition is true.

$$TR^{n+1}(f_i, f_j) = \text{inside, and } f_i.\text{action} \neq f_j.\text{action}$$

(d) Redundancy error: A filter f_i causes a redundancy error in f_j when any one of the following conditions is true.

$TR^{n+1}(f_i, f_j)$ = equal, f_i.action = f_j.action

$TR^{n+1}(f_i, f_j)$ = contains, f_i.action = f_j.action

(e) Redundancy warning: A filter f_i causes a redundancy warning in filter f_j when any one of the following conditions is true.

$TR^{n+1}(f_i, f_j)$ = inside, and f_i.action = f_j.action

$TR^{n+1}(f_i, f_j)$ = overlap, and f_i.action = f_j.action

By applying the above conditions for each $TR^{n+1}(f_i, f_j)$ and its action, we have found that f_1 causes a generalization warning in f_2, f_0 causes a redundancy warning in f_2, and f_0 causes a correlation warning in f_4. The filter f_3 is a disjoint filter and the filters f_0, f_1 are no error and warning filters. Thus, our system can successfully detect and classify the conflicts into two types of errors, three types of warnings, and no error and warning filters.

2.3 A Topology-based Spatial Analysis

As we have discussed earlier, **our first approach is the topology-based spatial analysis for the conflict detection in firewall polices.**

According to Max J. Egenhofer, topological relationships are a subset of spatial relationships [1]. 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. It does not preserve distances and directions. Topology refers to the way in which the filters are connected to each other.

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 figure 2.1 is represented spatially in Figure 2.3.

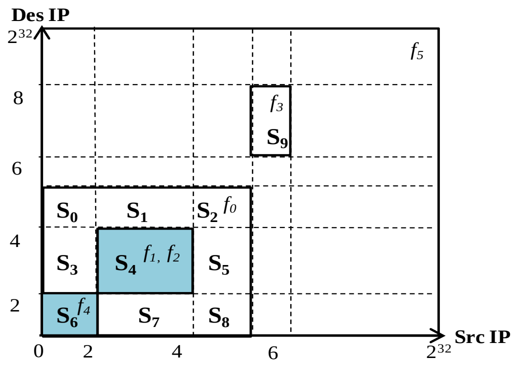


Figure 2.3 Spatial decomposition of filters of TFP1 defined in Figure 2.1

We perform a spatial decomposition of the n-dimensional space until dividing the filter boundaries reach its last dimension. The final decomposed space is called a subspace S_i. In a topology-based system, the location is discarded, and concentrates only on the uniqueness of the subspaces. In other words, **it only selects the subspaces with different filter sets and removes the subspaces with the same filter sets**. For example, even though, there are ten subspaces, $S_0 \ldots S_9$ resulting in the spatial decomposition from Figure 2.3, we consider only the unique subspaces for our topology-based spatial analysis. Therefore, we consider only the four unique subspaces with different filter sets, namely $\{\{f_0\}, \{f_0, f_1, f_2\}, \{f_0, f_4\}, \{f_3\}\}$ for extracting the TR^n of the filters.

2.4 Temporal Analysis

To find the conflicts in TFP, it is mandatory to temporally analyze the filters in the TFPs. We project the time constraints of the time-based filters in the time domain by projecting the filters in the temporal space and find the relationship between the filters $TR_{n+1}(f_i, f_j)$. **Our second approach is the geometry-based temporal analysis and the third approach is the topology-based temporal analysis. In both analyses, the filters are analyzed in temporal space to extract the $TR_{n+1}(f_i, f_j)$ of the filter sets.**

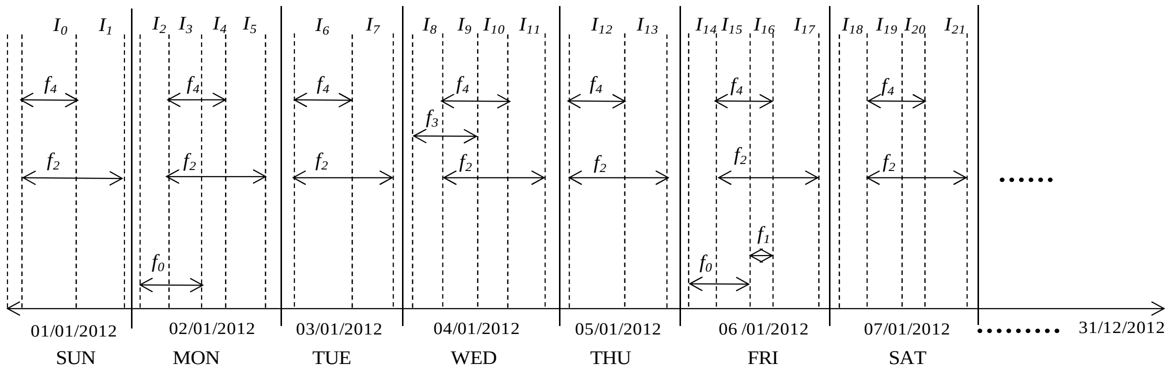
2.4.1 A Geometry-based Temporal Analysis

In this analysis, we define a conflict detection period CDP and a live filter as follows:

CDP: A CDP is a range of dates [Start, Stop], which specifies the filters to be analyzed in conflict detection.

Live Filter: A filter *f* is called a **live filter** if it is active during some time duration [t_1, t_2] such that CDP.Start ≤ t_1 ≤ t_2 ≤ CDP.Stop, and **a set of live filters LF** is taken into consideration in conflict detection.

However, for a given conflict detection period CDP, a periodic filter can be represented as a filter having a finite number of time periods. For example, for the given CDP = [01/01/2012, 31/12/2012], the temporal relationships among the filters in the TFP shown in Figure 2.1 are represented as shown in Figure 2.4. After the projection of the filters in the predetermined CDP, the filter sets present in each interval are extracted and analyzed to compute the TR_{n+1} of the filters. From this Figure, we can see that $TR_{n+1}(f_2, f_4)$ = contains, $TR_{n+1}(f_1, f_2)$ = inside, $TR_{n+1}(f_0, f_1)$ = disjoint, and so on. In this analysis, when the CDP is long, same number of filter sets repeats ample number of times and it takes huge computation time and memory to process all the filter sets for conflict detection. This drawback is driven away by the topology-based temporal analysis.



I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8 I_9 I_{10} I_{11} I_{12} I_{13} I_{14} I_{15} I_{16} I_{17} I_{18} I_{19} I_{20} I_{21}
f_4
f_3
f_2
f_1
f_0
……
01/01/2012
SUN
02/01/2012
MON
03/01/2012
TUE
04/01/2012
WED
05/01/2012
THU
06 /01/2012
FRI
07/01/2012
SAT
………
31/12/2012

Figure 2.4 Temporal Mapping of filters through geometry-based temporal analysis.

2.4.2 A Topology-based Temporal Analysis

In this analysis, we consider only the topology of the time-based filters projected in the temporal space. We propose a periodic cycle treatment which consists of a **mapping mechanism to treat the periodic filters**. It maps the time-based filters only on certain days and the period of temporal mapping is independent of the CDP. For example, if the conflict detection period is either a month or year, mapping of the periodic filters do not change and remains the same. The mapping depends on the input TFP and not by the CDP.

This mapping mechanism consists of three categories of mapping corresponding to the three types of filters. We denote the filters that are active in weekdays as FW, the filters that are active in every day as FE, and the filters that are active in date as FD. The three categories are 1) *specified days of the week*, *2) unspecified days of the week*, and *3) date*. The first one is for FW, the second one is for FE and the third one is for FD. In the *specified days of the week* mapping, the *FW and FE a*re mapped. It is performed to find the conflicting filters of FW in the input TFP. In the *unspecified days of the week,* as the name mentioned, the weekdays which are not specified in the input TFP are analyzed in which the FE is mapped. If all the weekdays are specified in the input TFP, then there is no need for mapping the *unspecified days of the week*. In *date* mapping, initially FE and FD are mapped. During mapping the filters in FD, the weekday of each filter is referred and the filters from FW that are active in that referred weekday is selected and mapped along with FD and FE.

For example, the filters in Fig.1 are mapped through the topology-based spatial analysis as shown in Figure 2.5. We can infer that, a single Monday and Friday is projected in the *specified days of the week* category. As the filters are active only on two weekdays in TFP, the filters active on everyday are mapped separately in the *unspecified days of the week* which corresponds to Tue, Wed, Thu, Sat and Sun.

In the *date category*, as the filter f_3 (04/01/2012) in Fig.1 falls on Wed, the *date* filters are mapped with filters active on Wednesday, and filters active on everyday to find the conflicting filters on 04/01/2012. In this mapping, there are 14 time intervals, and the filter sets present in all the intervals are processed to find the conflicting filters sets in TFP.

Thereby, the unnecessary repetitions of periodic filters are removed, as the mapping mechanism is independent of the conflict detection period.

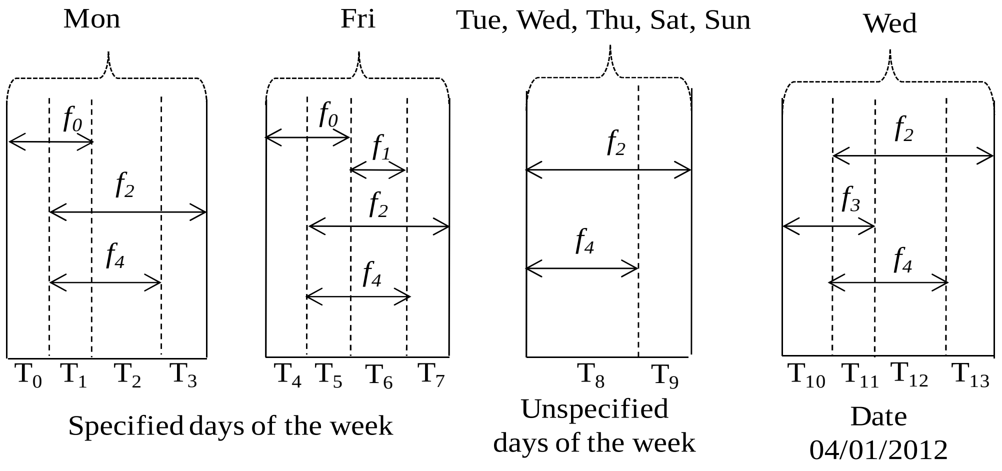


Figure 2.5 Temporal mapping of filters through periodic cycle treatment

2.5 A Spatiotemporal Analysis

To compute the conflicts in TFPs, it is necessary to analyze both the n-dimensional space and the temporal Space. Therefore we need to find methods how to combine both the analysis to find the $TR^{n+1}$of the filters. By combining n-dimensional spatial analysis, P1 and the temporal analysis P2 or P3, we can detect the conflicts in TFP by extracting the TR^{n+1} of the filters. We have proposed two architectures to combine the spatial analysis and the temporal analysis and they are called as Iterative and Simultaneous Analysis. In both the analysis, the temporal analysis can be performed either by P2 or P3.

2.5.1 Iterative Analysis

In this architecture, initially the filters are temporally projected in the temporal space either through geometry-based temporal analysis or topology-based temporal analysis. The projected filters are divided in their boundaries and the filter sets present in each interval are extracted. The filter sets in each interval, T_i are further analyzed in the n-dimensional space and their TR^n of the filters are computed. After finding TR^n corresponding to all the intervals, the results are compared and finally the TR^{n+1} of the filters are computed and the conflict classification is performed.

2.5.2 Simultaneous Analysis

In this architecture, the n-dimensional space and the temporal analysis are simultaneously analyzed. As we have mentioned earlier, the temporal analysis can be performed in both ways either through topology and geometry. We have explained the n+1-dimensional representation of the sample policy shown in Figure 2.1 in Figure 2.6. When the number of key fields in a packet header is '*n*', the packet arrival time and date is in the $(n + 1)^{th}$ dimension and a packet can be viewed as a point in an $(n + 1)$-dimensional packet space. As we have discussed earlier, the filter space of a time-based filter is represented by a subspace of an (n + 1)-dimensional packet space, which includes all the points of the packet that match the filter, including the time constraint. The filter space of the time-based filter f_i is represented as a cube in 3D and the example TFP is plotted in Figure 2.6, where the range of values in the $(n + 1)^{th}$ dimension is given by 00:00 06/01/2012 to 24:00 06/01/2012.

For a given packet P, time and date of the packet arrival T, and filter f_i, is Matched(P,T,f_i) does not become TRUE if is Active(f_i,T) is not TRUE. We identify the $TR^{n+1}(f_i, f_j)$ discussed above for each filter pair in the TFP by simultaneously analyzing the *n*-dimensional packet space and the temporal space to find the conflicts and its classification in the TFP.

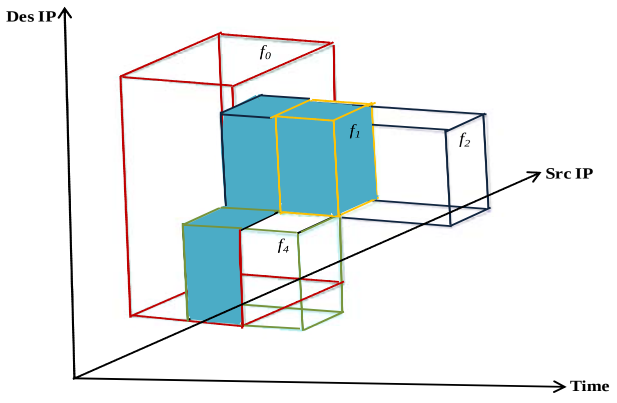


Figure 2.6 Projection of filters in 3D on 06/01/2012

2.6 Conclusion

In this chapter, we have discussed about the formalization of the TFP and the extraction of internal form of the TFP. We have also explained how conflict occurs in a TFP. We have explained how the n-dimensional spatial analysis and the temporal analysis can be combined and the two architectures of spatiotemporal analysis. We have found that by combining the three analyses in any two architectures, it becomes possible to find the TR^{n+1} of the filters and thus the conflict detection and classification can be done for any TFP.

Chapter 3

The System Proposal

3.1 System Overview

The main goal of the proposed system is to detect and classify the conflicts that can occur in TFP that consists of *m* filters and *(n+1)* number of key fields. It computes the TR^{n+1} of the filter pairs and classifies into errors and warnings for f_j according to the topological relationships. We have proposed an implementation method called BISCAL- a Bit-Vector Based Spatial Calculus to extract the TR^{n+1} of the filters and also we have proposed a time divisor which makes the temporal analysis. The time divisor can able to make both geometry and topology-based spatial analysis. The topology-based spatial analysis is made through the periodic cycle treatment. The overview of the proposed system is shown in Figure 3.1 and consists of a user interface, preprocessor, spatial divisor, time divisor, periodic cycle treatment, BISCAL and a conflict detector and classifier. 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. The system receives the TFP and follows the procedure given below.

Step 1: Initially, the network administrator makes a request for conflict detection for a TFP and inputs the TFP to the user interface. It converts the TFP into a text file format.

Step 2: The preprocessor translates the text file to an internal form of the TFP as we have explained in the chapter 2.

Step 3: In this step, the spatiotemporal analysis is performed. The TR^{n+1} of the filters are computed by analyzing the space and time, and the conflicts are detected and classified from the TR^{n+1} of the filters through BISCAL.

Step 4: The classified conflicts are displayed in the user interface and the results are given to the network administrator.

The step 3 is further explained in detail as follows:

Step 3.1: The key fields from *0 to n-1* are projected in *n*-dimensional space and the topology-based n-dimensional spatial analysis is performed.

Step 3.2: The nth key field is analyzed and the filters are projected in the temporal space and the temporal relationship, TR_{n+1} of the filters are computed. It can be achieved either by geometry-based temporal analysis or topology-based temporal analysis.

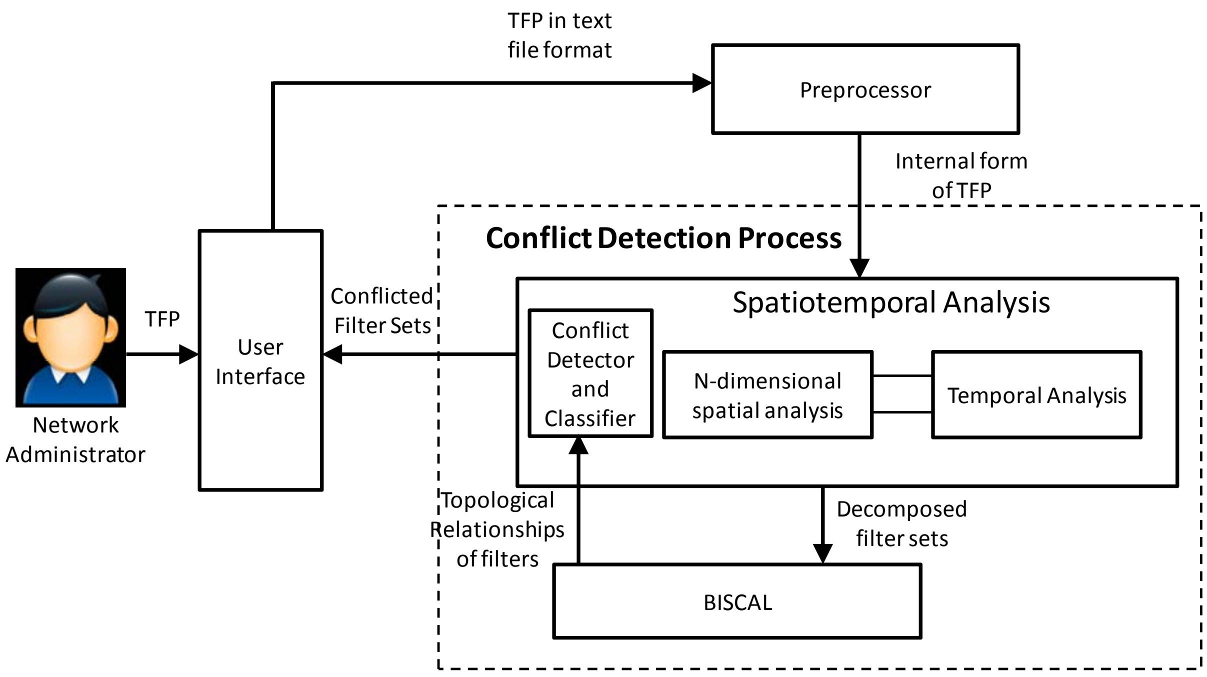


Figure 3.1 System Overview of the Conflict Detection System for TFP

Step 3.3: The n-dimensional analysis and the temporal analysis are combined either through simultaneous or iterative analysis and the decomposed filter sets are computed.

Step 3.4: The decomposed filter sets of the *0* to *n+ 1 key field* are given to the BISCAL, and the TR^{n+1} of the filters is extracted through BISCAL.

Step 3.5: Then, the conflict detector and classifier computes the conflicting filter sets from the TR^{n+1} of the filters. It classifies the conflicts as we have discussed in the previous chapter into two types of errors and three types of warnings.

3.2 Preprocessor

It consists of a text file reader and a translator which converts the input text file to the internal form which is the readable format in the conflict detection system. All the predicates in the TFP are converted into range values as we have explained in the chapter 2 and are called an internal form of TFP. All the *m* filters are converted into its corresponding internal form in the preprocessor and the internal form of TFP is fed to the spatiotemporal analysis.

3.3 N-dimensional Spatial Analysis

The spatial decomposition of n-dimensional space is performed until each filter is divided at all its boundaries in the last dimension. As we have discussed in the previous chapter, when the number of keys fields in a packet is *n*, the packet can be represented as a point in an *n*-Dimensional space called the packet space and the filter is represented by a rectangle in 2D and a hypercube in n-dimension. The filters are projected in n-dimensional space and decomposed in their boundaries and forms subspaces. The unique subspaces in the n-dimensional space are extracted and the filter sets present in each subspace are extracted.

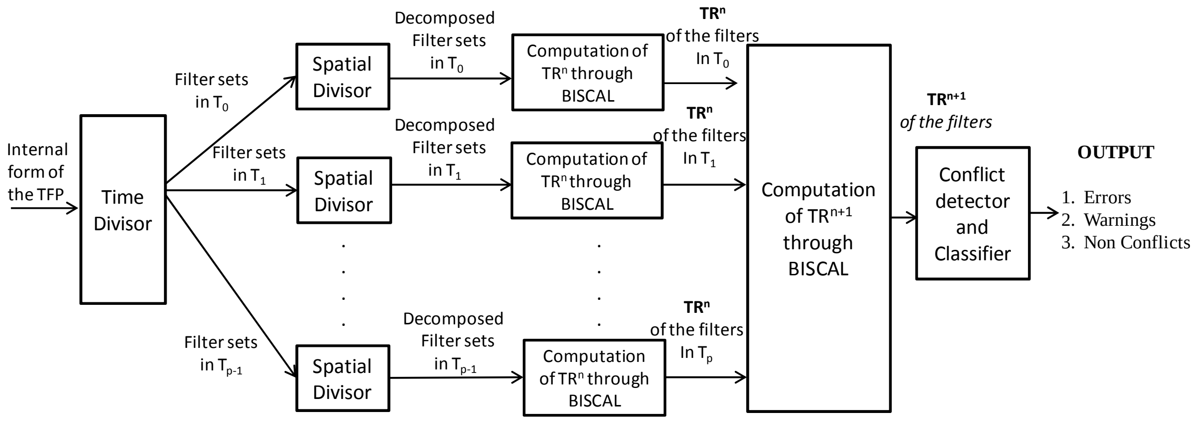
3.4 Temporal Analysis

We have proposed time divisor to implement the temporal analysis. In this analysis, the filters are projected in temporal space and the filters are divided in their boundaries and the filter sets present in each time interval are extracted. The mapping of the filters can be done either by geometry-based temporal analysis or the topology-based temporal analysis. In the geometry-based temporal analysis, the filters are projected in a predetermined conflict detection period, CDP and the filters are mapped along it. For example, if we want to detect conflicts for the year 2012, the CDP would be {[01/01/2012, 31/12/2012]}. In this period, the filters are projected and divided into intervals and the filter sets present in all the intervals are extracted.

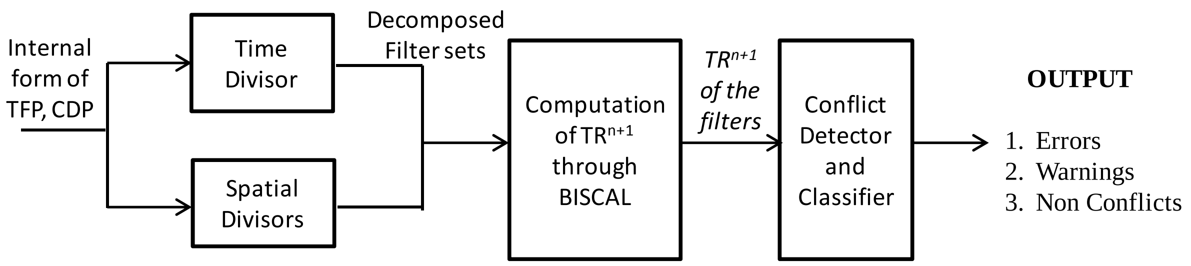
In the topology-based temporal analysis, the mapping of the filters in the time domain is independent of the CDP. The mapping period is determined by a mapping mechanism which has a periodic cycle treatment. It maps the filters only in a specified period consisting of few days. The filters are projected only in that certain days and the filters are divided in their boundaries and the filter sets present in all the intervals are computed and the decomposed filter sets are extracted by the time divisor.

3.5 Spatiotemporal Analysis

It combines the n-dimensional spatial analysis and the temporal analysis. We have proposed two architectures to combine the *n*-dimensional space and the temporal space. They are iterative and simultaneous analysis and their system overview is shown in Figure 3.2 (a) and Figure 3.2 (b) respectively.


Internal form of the TFP
Time Divisor
Filter sets in T_0
Filter sets in T_1
Filter sets in T_{p-1}
Spatial Divisor
Spatial Divisor
Spatial Divisor
Decomposed Filter sets in T_0
Decomposed Filter sets in T_1
Decomposed Filter sets in T_{p-1}
Computation of TR^n through BISCAL
Computation of TR^n through BISCAL
Computation of TR^n through BISCAL
TR^n of the filters In T_0
TR^n of the filters In T_1
TR^n of the filters In T_p
Computation of TR^{n+1} through BISCAL
TR^{n+1} of the filters
Conflict detector and Classifier
OUTPUT
1. Errors
2. Warnings
3. Non Conflicts

(a) Iterative Analysis


Internal form of TFP, CDP
Time Divisor
Spatial Divisors
Decomposed Filter sets
Computation of TR^{n+1} through BISCAL
TR^{n+1} of the filters
Conflict Detector and Classifier
OUTPUT
1. Errors
2. Warnings
3. Non Conflicts

(b) Simultaneous Analysis

Figure 3.2 Two architectures of spatiotemporal analysis

3.5.1 Iterative Analysis

The system overview of this analysis is shown in Figure 3.2(a). Initially, the filters are decomposed in the time domain. The filters are projected in the temporal space and decomposed into boundaries and forms time intervals, T_i. For example, if there is 'p' number of intervals in the temporal space, then the filter set present in each interval are extracted. The filters sets present in each interval are further analyzed in *n*-dimensional space to find the TR^n of the filters through BISCAL. For each interval, the TR^n is computed and the results are fed to the next step. All the interval results are combined and the TR^{n+1} of the filters are computed through BISCAL. The conflicting filter sets of the TFP are computed from the TR^{n+1} of the filters and conflicts are further classified in the conflict detector and the classifier.

3.5.2 Simultaneous Analysis

The space and time are simultaneously analyzed in the *n+1*-dimensioanl space and the decomposed filter sets in the *n+1*-dimensional space are given to BISCAL as shown in Figure 3.2(b). It computes the TR^{n+1} of the filters and the results are fed to the conflict detector. In the conflict detector and classifier, the conflicts are computed from the TR^{n+1} of the filters and classified as shown in the previous chapter.

3.6 BISCAL

We have proposed a Bit-vector based Spatial Calculus named BISCAL and constructed a new conflict detection framework through BISCAL which analyzes the TR^{n+1} of the filters. It is a systematic and simple implementation method to find the TR^{n+1} of the filters. The main advantage of BISCAL is that the operations on the filter sets become easier. It consists of seven primitive operators and three Characterization Vectors, CVs. It converts the filter sets into bit-vector format and computes the TR^{n+1} of the filters through the seven primitive operations. The characterization vectors are used to preserve the intermediate results.

A TFP which consists of *m* filters is represented by a bit-vector [$b_0 ... b_{m-1}$], where a bit b_i in the bit-vector represents the filter f_i. If a filter is selected from the TFP, then the value of the corresponding bit is 1, and if not, then the value of the bit is 0. For example, let us consider a TFP consist of five filters f_0, f_1, f_2, f_3 *and* f_4. If filters (f_1, f_3, f_4) are selected from TFP, then the bit-vector is [01011]. In this method, we make the bit-vector for the decomposed filter sets for all the 0 to n key fields and then the TR^{n+1} of the filters are computed. Because this thesis focuses on bit-vectors, hereinafter we will refer to a bit-vector as simply vector.

3.7 Conclusion

We have explained the overall system overview and each component is further explained in detail in the upcoming chapters. Our proposed system computes the $TR^{n+1}$of the filters through BISCAL by simple logical operations and the implementation of BISCAL is explained in detail. By implementing our system proposal, the network administrator could take decisions in removing the conflicts in TFPs and can easily manage the TFPs in industrial environments.

Chapter 4

A Topology-Based Spatial Analysis through BISCAL

We have proposed a topology-based spatial analysis for the firewall policies to detect conflicts in the firewall policies. We introduce a bit-vector based spatial calculus named BISCAL and constructed a new conflict detection framework through BISCAL which analyzes the TR^{n+1} of the filters to detect and classify conflicts. In this chapter, we concentrate only on the *n*-dimensional spatial analysis and therefore we discuss the extraction of TR^n of the filters through BISCAL. We discuss the design and implementation of the topology-based conflict detection system that shows how effectively the BISCAL can be adopted for conflict detection. In this chapter, we discuss the extraction of the TR^n of the filters and therefore we ignore the time-field in the implementation of the sample TFP shown in Figure 2.1 and therefore the input of the conflict detection system becomes an FP.

4.1 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 any dimension.

4.1.1 Data Structure: Bit-Vector

An FP which consists of *m* filters is represented by a bit-vector $[b_0 ... b_{m-1}]$, 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 1s in the vector to its corresponding filters. For example, **V2S** ([11100]) = {f_0, f_1, f_2}.

4.1.2 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}}.

4.1.3 Characterization Vectors

The characterization vectors are the vectors that characterize the topological relationship of the filters. There are two types of characterization vectors: 1) vectors characterizing 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.

A. 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.

a) 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 2.3, f_0, f_1 and f_2 co-exist in S_4, therefore the vector representation is, OV= [11100].

b) 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 2.3, f_1, f_2 and f_4 are fully covered in two-dimension and therefore, FV= [01101].

c) 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 2.3, f_3 is a disjoint filter and therefore DV= [00010].

B. 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 $X_i(f_0,...,f_{m-1})$ 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 4.1 shows an example of the spatial division for the projection of the filters in FP on the 0th axis corresponding to the 0th key X_0 or SrcIP and shows that six intervals, $I_0 \ldots I_5$, are made by the decomposition of $X_0(f_0 \ldots f_5)$.

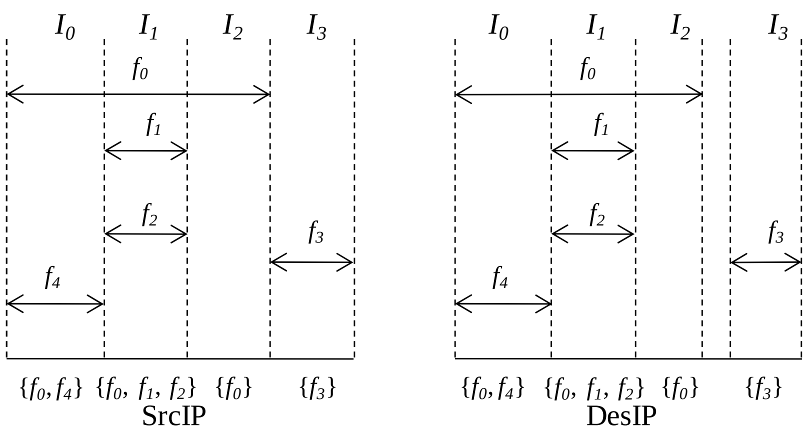
Each interval has a set of filters, called sub-domain filter set, in which *ith* predicate of the filter is always true within the interval. The sub-domain filter sets for all the intervals on the *ith* axis except the empty sets form a set, named SFS_i. Each sub-domain filter set is transformed into 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_0, f_1, f_2}, {f_0, f_4}, {f_3}} and SVS_0 = {[10000], [11100], [10001], [00010]} for the spatial division for the projection of the filters in sample FP mentioned in the above.

PCV_i characterizes the topological relationship of the projected filters on the *ith* axis and consists of the following three kinds of vectors.

a) $POVS_i$: 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 *ith* axis of the packet space. $POVS_i$ is a set of all $POVS_i$s. For example, in figure 4.1, $POVS_0$ = {[11100], [10001]}.

b) 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 *ith* axis of the packet space. For example, in figure 4.1, the projections of {f_1, f_2} and {f_4} are fully covered and therefore, PFV_0= [01101].

c) PDV_i: If a filter f_i is disjoint from all other filters projected on *ith* axis of the packet space, then the value of b_i is set to 1 in PDV_i. For example, in figure 4.1, f_3 is disjoint and therefore PDV_0= [00010].



I_0 I_1 I_2 I_3
f_0
f_1
f_2
f_3
f_4
$\{f_0, f_4\}$ $\{f_0, f_1, f_2\}$ $\{f_0\}$ $\{f_3\}$
SrcIP
I_0 I_1 I_2 I_3
f_0
f_1
f_2
f_3
f_4
$\{f_0, f_4\}$ $\{f_0, f_1, f_2\}$ $\{f_0\}$ $\{f_3\}$
DesIP

Figure 4.1 Spatial division of filters

4.2 Implementation of a Topology-Based Spatial Analysis through BISCAL

4.2.1 System Overview

Our 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 a 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 chapter 2.

The overview of the proposed system is shown in figure 4.2 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 ith predicate of the filters in FP and represents projections of the filters in FP on the ith axis of the packet space, $X_i(f_0,...,f_{m-1})$, where i is from 0 to n-1.

STEP 2: The spatial divisor makes SFS_i by the spatial division of the projection $X_i(f_0,...,f_{m-1})$

on the ith axis in the way as described in the previous section 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 previous section 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 TR^n 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 the chapter 2.2.

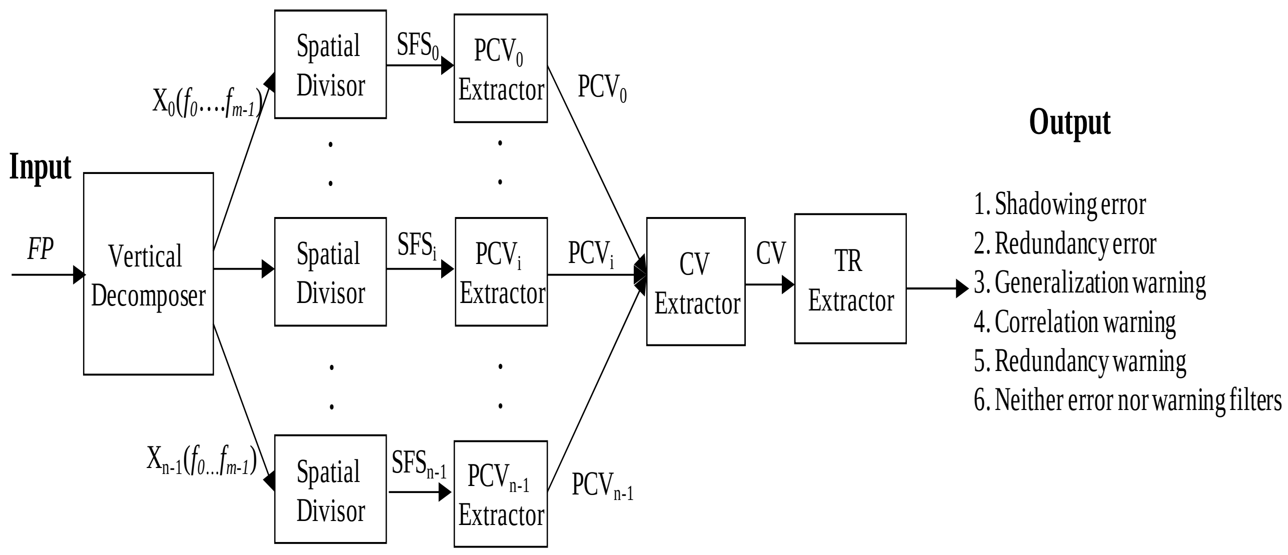


Figure 4.2 System overview of topology-based conflict detection system through BISCAL

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.

4.2.2 PCV Extractor

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

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

Initialize $POVS_i$= {Ø};

For all v SVS_i, if C1 (v) > 1, $POVS_i$=$POVS_i$ v;

For example, $POVS_0$ = {[11100], [10001]} 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 ([11101]) = [00010].

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 4.1, f_0 is a non-fully covered filter, because f_0 is alone in many subspaces. 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 = {Ø};

For all v SVS_i, if C1 (v) =1, NF_i = NF_i v;

PFV_i = NOT (OR (NF_i));

For example, in X_0, SVS_0 = {[10000], [11100], [10001], [00010]}, as mentioned above, NF_0 = {[10000], [00010]}, and PFV_0= NOT ([10010]) = [01101].

4.2.3 CV Extractor

CV Extractor calculates the CVs using the PCV_i(i=0...n-1), calculated in the previous subsection. Before the computation of the CVs, we remove the disjoint filters from $POVS_i$(i=0…n-1) and PFV_i(i=0...n-1) to discard the disjoint filters from conflict detection. Using this technique, we improve our system efficiency, as unnecessary computations are removed by discarding 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_i', 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_i' are calculated as follows. In this step, the disjoint filters are removed from OVS_i' and FV_i' to discard the possibility of a disjoint filter in the upcoming conflict detection steps.

$DV' = OR\ (PDV_0 ... PDV_i ... PDV_{n-1})$;

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

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

$OVS = \{Ø\}$;

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

For i = 2 to n-1, repeat $IR_i = C\text{-}AND\ (OVS_i, IR_{i-1})$;

For all $v ∈ IR_{n-1}$, if $C1\ (v) = 1$, $OVS = OVS ∪\ v$;

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:

$NF_n = \{Ø\}$;

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

FV= AND (FV_0'...FV_{n-1}', (NOT (OR (NF_n))));

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

DV= NOT (OR (OVS));

For example, the CVs for SAMPLE TFP, shown in Table 2.1, are computed for two dimension (SrcIP, DesIP), as follows: OVS = {[11100], [10001]}, FV = [01101] and DV = [00010].

4.2.4 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 Chapter 2.2. Filter pairs that have the conflict causing topology are defined in CCTP as follows:

CCTP = {(f_i, f_j) | TR (f_i, f_j) {inside, contains, equal, overlap}}.

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

$CCTP_{topology}$ = {(f_i, f_j) | TR (f_i, f_j) = topology}, where 'topology' is one of inside, contains, equal, or overlap.

For example: $CCTP_{overlap}$ = {(f_i, f_j) | TR (f_i, f_j) = overlap}.

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** ([00010]) = {f_3}, which shows that f_3 is a disjoint filter in the sample TFP.

Step 5-2: Computation of CCTP

CCTP = {Ø};

For each v OVS, CCTP = CCTP ***PF*** (v);

For example, the CCTP of the sample FP is as follows: OVS= {[11100], [10001]}, CCTP = {(f_0, f_1), (f_0, f_2), (f_1, f_2), (f_0, f_4)}.

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 = {Ø};
S = V2S (FV);
for each $f_k$ of S, do {
        $FF_k$ = [11...1];
        for each v ∈ 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 the sample FP is calculated as follows:

S= V2S ([01101]) = {f_1, f_2, f_4} and OVS= {[11100], [10001]}. Then, FF_1 = AND {[11111], [11100]} = [11100].

Similarly, FF_2 and FF_4 are calculated as follows: FF_1 =AND ([11111], [111000]) = [11100], FF_3 =AND ([11111], [10010]) = [10010].

Set f_k bit to '0' in FF_k, and therefore, FF_0= [01100], FF_1= [10100] and FF_3= [10000].

SP= PO ({f_1}, {f_0, f_2}) ∪ PO ({f_2}, {f_0, f_1}) ∪ PO ({f_4}, {f_0}) = {{f_0, f_1, f_2}, {f_4, f_0}}.

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.

$CCTP_{equal}$ = {(f_i, f_j) | f_i, f_j ∈ S, (f_i, f_j) ∈ SP};

$CCTP_{inside}$ = {(f_i, f_j) | f_i ∉ S, f_j ∈ S, (f_i, f_j) ∈ SP};

$CCTP_{contains} = \{(f_i, f_j) \mid f_i$ □ S, f_j $\underline{□}$ $S, (f_i, f_j)$ □ $SP\}$;

The $CCTP_{overlap}$ is calculated as follows:

$CCTP_{overlap} = CCTP$□SP;

For example, the SP for the sample FP calculated in the STEP 5-3-1 leads us to the following two sets: $CCTP_{equal} = \{(f_1, f_2)\}$ and $CCTP_{inside} = \{(f_0, f_1), (f_0, f_2), (f_0, f_4)\}$.

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 chapter 2.2, any filter f_j is classified. For example, the filters in the sample TFP in Figure 2.1 are classified as follows: f_1 has a shadowing error caused by f_0, f_2 has a redundancy error caused by f_0, f_4 has a shadowing error caused by f_0 and f_0 and f_3 are no error and warning filters.

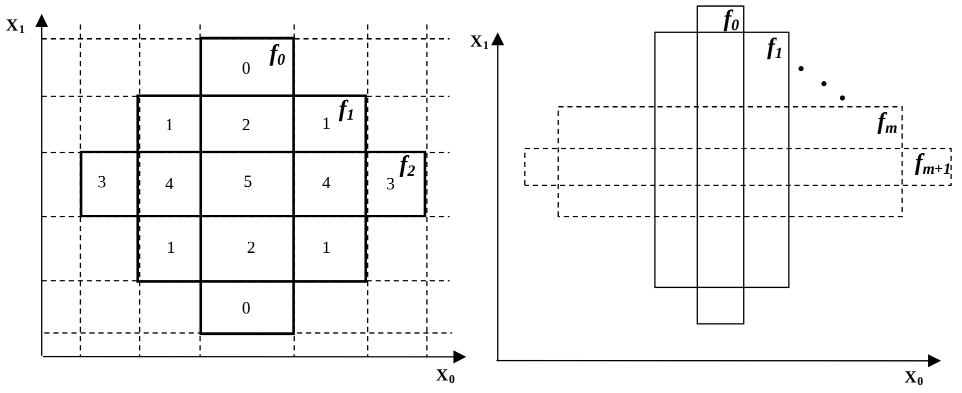
4.3 Evaluation

We have evaluated the n-dimensional spatial analysis for conflict detection for FPs using a mathematical analysis with a special case of FP. We have also developed a prototype of our system in JAVA programming language and then performed an experimental analysis to evaluate the efficiency of the conflict detection for FPs through a topology-based n-dimensional spatial analysis.

4.3.1 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 4.3(a) and *m*+1 filter is shown in figure 4.3(b).

In figure 4.3(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 4.3 (a).



(a) 3 filters (b) m+1 filters

Figure 4.3 FPA in 2-dimension

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 4.3 (b), we have derived the difference in subspaces required for the topology and geometry approaches for figure 4.3.

Table 4.1 Results of Mathematical analysis

Number of filters	Geometry	Topology
2	5	3
5	41	15
100	19801	5050
1000	1998001	500500

When an m+1th 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+1th filter is added, the number of subspaces considered for conflict detection is m+1. However, in the geometrical approach, when the m+1th 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:

$NS_i(m) = (m^2+m)/2$ and

$NS_i(m+1) = NS_i(m) + (m+1)$.

Geometry:

$NS_i(m) = m^2 + (m-1)^2$ and

$NS_i(m+1) = NS_i(m) + 4(m)$.

We have substituted different values of m, and tabulated the number of subspaces in Table 4.1. 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 verified it using the experimental analysis in the next section.

4.3.2 Experimental Analysis

We have evaluated our system by performing a comparative analysis between the proposed system and conventional geometry-based approaches [4, 9]. 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 conflict detection techniques [4]-[26], 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 [4, 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 4.4. It is clear from the graph that our proposed topology-based system performs better than the geometrical approach.

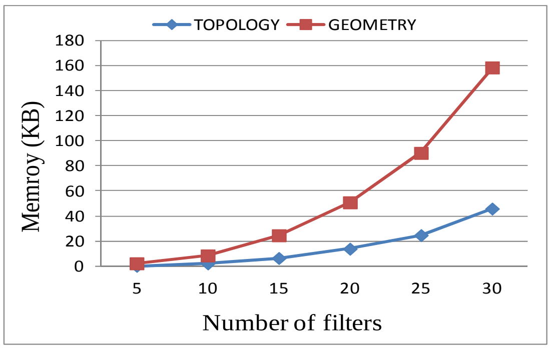
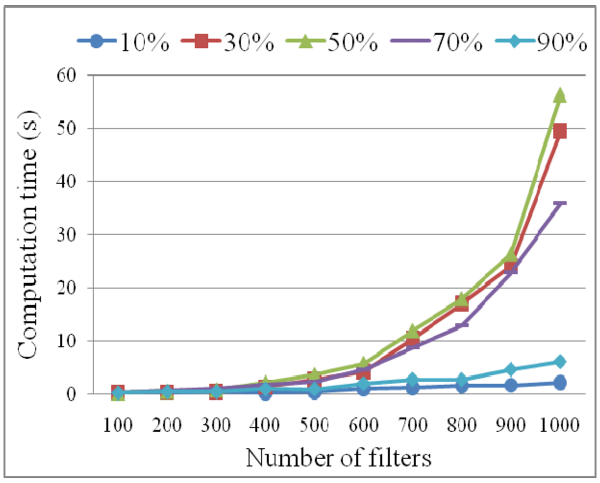
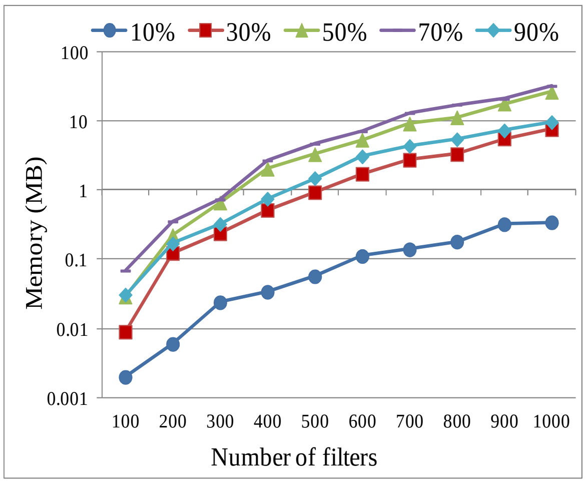


Figure 4.4 Comparison of topology and geometry

Exp.2 is performed by varying the ratio of the wildcards of the input filters, as shown in Figure 4.5 (a) and Figure 4.5 (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.

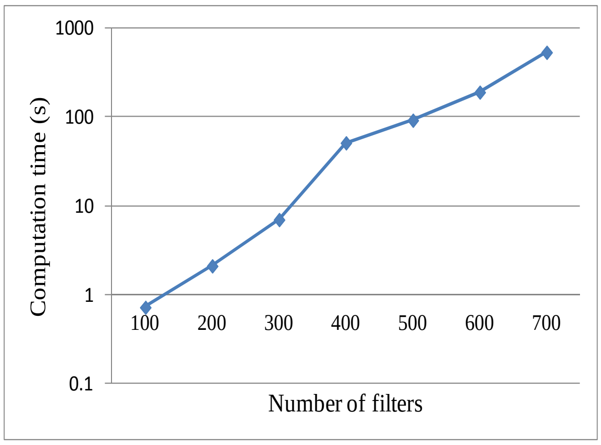




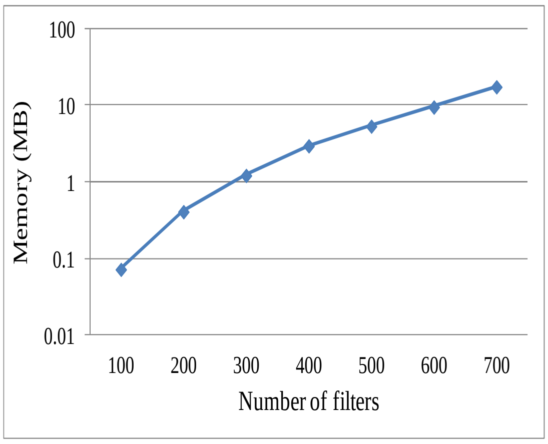
(a) Computation Time (b) Memory

Figure 4.5 Performance Analysis of changing the ration of wildcard in FPB

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 Figure 4.6 (a) and Figure 4.6 (b). For example, the system takes only 100 seconds to detect and classify the conflicts for m = 500.



(a) Computation time



(b) Memory

Figure 4.6 Performance Analysis of system using FPB

4.4 Conclusion

In this chapter, we have completely discussed about the extraction of conflicts for a FP. By the above explanation, it is very clear that BISCAL is a simple and systematic way of extracting the conflicts through simple logical operations and characterization vectors. The main advantage of the topology-based approach is only the unique subspaces are considered for conflict detection and therefore the computation time and memory required is reduced. In addition, implementation through BISCAL preserves the intermediate results in the characterization vectors and also the disjoint filters are removed in the intermediate stage itself and improve the efficiency of the conflict detection system. We have also proved in the mathematical analysis with a special case that topology-based approach performs better than geometry.

Chapter 5

Temporal Analysis

Conflict detection in TFPs requires the analysis of time-domain for finding TR^{n+1} of the filters. We know that by ignoring the time-field, false positive results occur saying a non-conflict as a conflict and thereby the workload of the network administrator becomes too hectic. In our research we have extensively studied the possibilities of analyzing the time-field in temporal space through geometry and topology. We have discussed the algorithms and steps to compute the temporal analysis through two methods, geometry-based and topology-based. The temporal analysis is been implemented by **time-divisor** which can perform the temporal analysis by either of the two analysis. As we have discussed earlier, we represent the filters that are active in weekdays as FW, filters that are active in every day as FE and the filters that are active in dates as FD.

5.1 A Geometry-based Temporal Analysis

In this analysis, the filters are projected in temporal space for a predetermined CDP and the decomposed filter sets, SFS_n in the time field are computed. The computation of CDP and live filter is discussed in the Chapter 2.4.1.

For a given CDP, the time divisor discovers the decomposed filter sets of the filters. The input is the $X_n\{f_0,\dots f_{m-1}\}$ which represents the time field of the TFP that obtains the values in the ActTIME fields of the filters and outputs the decomposed filter sets, SFS_n according to the following steps.

STEP1: It adds all the periodic filters in the TFP to a set of live filters LF and selects a non-periodic filter f_i from the TFP that satisfies the following predicate and adds it to LF: CDP.Start ≤ f_i.ActTIME.DAY ≤ CDP.Stop.

STEP2: It maps a filter in LF onto the temporal axis, i.e., the $(n + 1)$th axis, according to the values in the time field of the filter, i.e., start and stop f_i.ActTIME. For example, if a filter f_i is active on Wednesday from 08:00 to 12:00, it is mapped from 08:00 to 12:00 on every Wednesday in the CDP. If a filter is active on every day, the boundary values corresponding to all the days in the CDP are identified.

STEP3: It decomposes each filter in its boundaries, which are mapped onto the temporal axis by in STEP2, so that time intervals such as I_0, I_1, etc., are created similar to those in the spatial divisor discussed in the previous chapter.

STEP4: It obtains a set of filters from each interval created in STEP3 and makes SFS_n using the filter set.

STEP2 is described in detail as follows.

STEP2.1: It classifies the filters in LF into two groups: FW and FD, where FW is a set of periodic filters and FD is a set of non-periodic filters.

STEP2.2: It makes a set of dates: TDS, which are within the CDP.

STEP2.3: It makes an array of a set of boundary values, L, by executing the **FilterMapper** algorithm shown in below, where L[i] is the set of boundary values [X, Y) of the filter f_i, where X.date and X.time are the date and time for the start edge of a time interval and Y.date and Y.time are the date and time for the stop edge of the time interval.

In the **FilterMapper**, the function GetDay(date) translates the given date to a day of the week.

[FilterMapper]

```
for each date in TDS do {
 for each $f_i$ in LF do{
      if ((($f_i$  FW) and  (GetDay(date)  $f_i$.ActTIME.DAY)) or  (($f_i$  FD) and  date = $f_i$.ActTIME.DAY))) then {
            X.date = date; X.time = $f_i$.ActTIME.TIME.start;
            Y.date = date; Y.time = $f_i$.ActTIME.TIME.stop;
            append (X, Y) to L[i];}}}
```

For example, the computational steps for the TFP shown in Figure 2.1 where CDP = [01/01/2012, 31/12/2012] are as follows: step 1 adds all the periodic filters f_0, f_1, f_2, f_4 into **LF** and selects the non-periodic filter f_3 and adds it into the **LF**, since 01/01/2012 ≤ 04/01/2012 ≤ 31/12/2012. Step 2 computes the boundary values of the filters in the CDP. In this step, the **FilterMapper** algorithm finds that the non-periodic filter f_3 has a single boundary value: ([04/01/2012, 08:00], [04/01/2012, 12:00]) and that the filter f_0 has multiple boundary values: (([01/01/2012, 08:00], [01/01/2012, 12:00]),...,([31/12/2012, 08:00], [31/12/2012, 12:00])). After determining the boundary values, the filters are mapped in the temporal axis accordingly. In step 3, the mapped filters are decomposed and intervals such as I_0, I_1, etc., are formed as shown in Figure 2.4. Due to space limitations, we have shown the filters in the temporal space only from 01/01/2012 to 07/01/2012 in Figure 2.4. In the final step, the filter sets in each interval are added to SFS_2 and therefore, $SFS_2 = \{\{f_2, f_4\}, \{f_2\}, \{f_0\}, \{f_0, f_2, f_4\}...\}$.

In this analysis, when the CDP is long, the periodic filters repeat numerous times and therefore the number of decomposed filter sets would be huge. For example, as we have seen temporal mapping in Figure 2.4, the filters in FE repeats nearly 365 times in a year and the filters in FW repeats nearly 52 times in the mapping. The number of filters sets in all the intervals will be huge, and takes huge computation time and memory to find the conflicts. To solve this problem, we have proposed a topology-based temporal analysis to discard the unnecessary repetitions of the periodic filters.

5.2 A Topology-based Temporal Analysis

In this topology-based temporal analysis, rather than analyzing the geometrical location of the time-based filters in the CDP, we only focus on the topology of the filters. To detect the conflicts, it is necessary to compute the temporal relationship of the filters. Therefore, the filters are mapped only in certain days, and the decomposed filter sets, SFS_n are computed. By this approach, the unnecessary repetitions of the periodic filters can be eliminated. Therefore we have proposed a mapping mechanism in for treating the periodic cycle filters.

In this analysis, the time divisor consists of seven primitive operations to achieve the mapping mechanism and they are Cycle Separator, Divide, Select-1, Select-2, Sum, Decompose and Remove. We briefly discuss the necessity of each operation below. In order to map filters in their corresponding days, initially filters are separated using **cycle separator**. As we map the filters corresponding to day-basis, filters longer than a day is divided using **divide** operation. **Select-1** operation selects the filters that are active in weekdays, FW and **Select-2** operation selects the filters that are active in everyday, FE and the filters that are active in date, FD. The different types of filters, FE, FW and FD are mapped together using **sum** operation. Filters are decomposed in filter boundaries and SFS_n are extracted using **decompose** operation. The unnecessary filter sets are removed using **remove** operation. We have shown the mapping of topology-based temporal analysis in figure 2.5 and the flow of operations of the time divisor is shown in Figure 5.1.

5.2.1 Primitive Time Handling Operations

We explain the primitive operations with two types of filter sets where F_i is set of filters and $\mathcal{F}_i$ is set of filter sets and the operations are explained for the sample TFP given in the Figure 2.1

1. *Cycle Separator:* It separates the input filter based on the type of filter. It separates the filters into *day*, *day of the week* and *date* filters. Cycle Separator (F) = (FD, FW, FA). For example, *Cycle Separator* ({f_0, f_1, f_2, f_3, f_4}) = {{f_2, f_4},{f_0, f_1}, {f_3}}.

2. *Divide:* If any of the given input set of filters is longer than a day, it is divided into many single-day filters. Input: F_i Output: F_j where F_i holds the input filter sets and F_j holds the output filter sets. For example, *divide* (f_0, f_1) = (f_0', f_0'', f_1). It divides f_0 into f_0'(active on Mon) and f_0'' (active on Fri). The other filters are returned as like the input.

3. *Select-1:* It selects the filters based upon the specific day of the week. Select-1(F_i) = (F_{sun},…,F_{sat}), where F_{sun} is the set of filters that are active in any Sunday to F_{sat} is the filters that are active on Saturday. If there are no filters on that particular day of the

week, it returns the empty set. For example, *Select-1*({f_0', f_0'', f_1}) = ({},{f_0'}, {}, {}, {}, {f_0'', f_1}, {}).

4. *Select-2:* It selects the filter sets based upon the weekday and date. For example, for Figure 2.1, *Select-2*({f_3}) = ({}, {}, {}, {f_3}, {}, {}, {}). The difference between *Select-1* and *Select-2* operation is, if multiple filters fall on Sunday with a different date, the filters are returned separately.

5. *Sum:* It computes the union of all the inputs. The input can be either F_i or $\mathcal{F}_i$ *Sum* ({F_i}, {F_j}) = ({F_i, F_j}), SUM ({$\mathcal{F}_i$}, {$\mathcal{F}_j$}) = ({$\mathcal{F}_i$,$\mathcal{F}_j$}). For example, For Fig.1, *Sum* ({f_0}, {f_1, f_3, f_4}) = {f_0, f_1, f_3, f_4}.

6. *Decompose:* It decomposes the input filters in their boundaries and returns the filter sets which correspond to each interval, T_i. Input: F_i, Output: $\mathcal{F}_i$ For example, in Figure 2.5, when the *unspecified day* filters are decomposed, then, *decompose* ({f_2, f_4}) = {($f_2 f_4$), (f_2)}.

7. *Remove:* This operation removes the alias filter sets. Input: $F_{i\text{-}i}$, Output: F_{ij}. For example, *Remove {{$f_0f_1f_3$}, {f_0f_3}, {$f_0f_1f_3$}, {f_0f_3}, {$f_0f_2f_3$}} = {{$f_0f_1f_3$}, {f_0f_3}, {$f_0f_2f_3$}}.*

5.2.2 Computation of SFS_n

The flow of operations for the mapping mechanism for the topology-based temporal analysis is shown in Figure 5.1. The input is the X_n{f_0,...f_{m-1}} which represents the time field of the TFP. Initially, the time-based filters are separated using *cycle separator*. It results in three set of filters where FD = {f_2, f_4}, FW' = {f_0, f_1} and FD' = {f_3}. In Figure 5.1, the numbered circles corresponds to the operations performed in mapping unspecified days, specified days and date filters respectively. If any filter is longer than a day, the filters are divided using *divide* operation. The filter f_0 is divided into two filters f_0', active on Mon and f_0'' which is active on Fri.

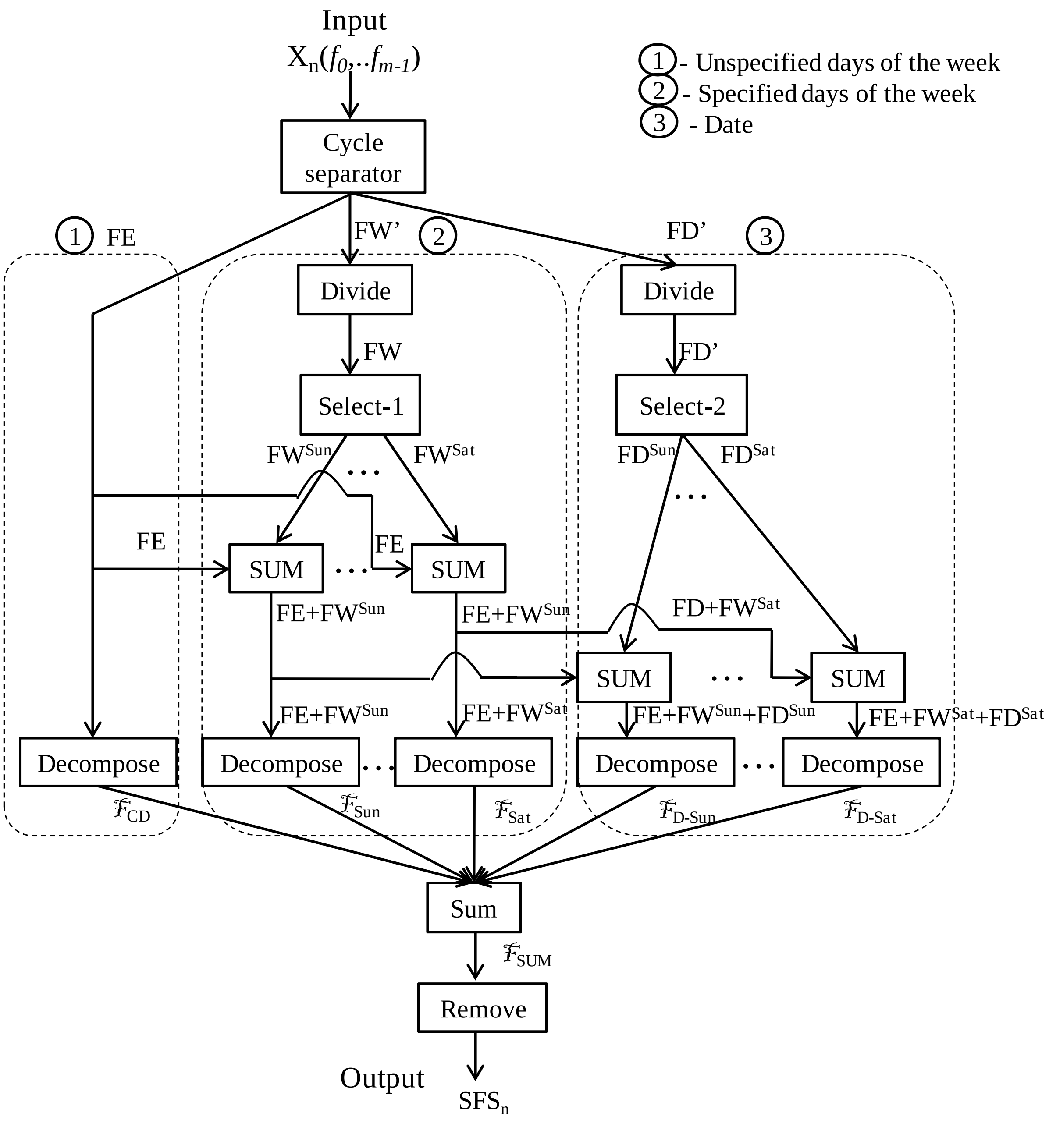


Figure 5.1 Flow of operations in the mapping mechanism

In the next step, the FW and FD undergo *select-1* and *select-2* operation respectively. For the sample TFP in Figure 2.1, it selects the filters that are active in different weekdays are, $FW^{Mon} = \{f_0'\}$, $FW^{Fri} = \{f_0''\}$ and $FD^{Wed} = \{f_3\}$. The FE is mapped with FW using *sum* operation. Likewise, FD is summed with FE and a subset of FW (the filters that are active in the weekday of FD) is summed using *sum* operation. The remaining flow of operations is shown only for Sun and Sat as all the other days of the week are similar. The next step is decomposing the filters in their boundaries and the filters in each T_i are extracted. As per the computational steps, F_{CD} = {{f_2,f_4}, {f_2}}, F_{Mon} = {{f_0'}, {f_0'f_2f_4}, {f_2f_4}, {f_2}}, F_{Fri} = {{f_0''}, {f_0''f_2f_4}, {$f_1f_2f_4$}, {f_2}}, $F_{D\text{-}Wed}$ = {{f_3}, {$f_2f_3f_4$}, {f_2f_4}, {f_2}}. If all the seven days of the week are specified, then there is no need of *unspecified days of the week* separately. The filter sets identified in decompose operation are summed up using the *sum* operation. Finally, the alias filter sets are removed using *remove* operation. A filter which is active in multiple days are represented as *f'* and *f''* for showing the differentiation in the mapping. As far as conflict detection is concerned, there is no need of separation, therefore the multiple filters are reunited as before as a single filter. Therefore the alias filters of divided filter sets are also considered and removed in the remove operation if any. The filter sets for SFS_2= {{f_2f_4}, {f_2}, {$f_0f_2f_4$}, {f_0}, {$f_1f_2f_4$}, {f_3}}. Therefore, there exists six unique filter sets in the time-field that must be considered for conflict detection.

5.3 Conclusion

We have elaborately discussed the extraction of SFS_n in terms of geometry-based and topology-based analysis. We have found that topology-based temporal analysis performs better than the geometry-based, as the unnecessary repetitions are removed by projecting the filters in a short period of few days and as a result the number of filter sets considered for conflict detection is comparatively reduced.

Chapter 6

Spatiotemporal Analysis

Conflict detection in TFP is a challenging task for managing the firewall policies. Even though various conflict detection techniques were proposed, if they are treated with TFPs, they produce false positive results as the time-field is ignored. There is no significant research made in the field of conflict detection in TFPs. We have proposed a conflict detection system for TFPs which can be achieved by spatiotemporal analysis. We have developed two architectures to perform the spatiotemporal analysis as follows: (1) Simultaneous Analysis, (2) Iterative Analysis. Both of the analyses detects and classifies the conflicts in the given time-based firewall policy TFP, which consists of *m* filters and $(n + 1)$ fields. It computes the $(n + 1)$-dimensional topological relationship of each filter pair (f_i, f_j), $TR^{n+1}(f_i, f_j)$ and classifies it into errors and warnings for f_j according to the topological relationships. All the conflicting combinations of the filters are obtained from the CVs and classified into five types of conflicts and other results (no error and warning filters). We have also performed some evaluations to validate the performances of the different architectures. We have made experimental analysis to show the percentage of the false positive results that can occur when time field is ignorned, and have also performed evalutaions to find the better performance systems.

6.1 Simultaneous Analysis

In this architecture, the temporal analysis and the topology-based spatial analysis are simultaneously analyzed. The system overview is shown in the Figure 6.1. The main difference between the system overveiw of the n-dimensional spatial analysis and the simultaneous analysis is that an additional dimension (the n+1th dimension) for the time-domain is analyzed.

The system receives the internal form of the TFP and CDP as the input and follows the procedures given below: (1) The vertical decomposer divides the TFP into $(n + 1)$

divisions, in which each division includes the *i*th field of the TFP and is represented by X_i $(f_0 \ldots f_{m-1})$. (2) For each *i*th field from $i = 0$ to $n - 1$, the spatial divisor projects the filters on the *i*th axis and decomposes them in their boundaries to form intervals, and the filter sets in each interval are added to the set of filter sets for the *i*th field SFS_i. The time divisor finds the interaction of the filters in the time field X_n $(f_0 \ldots f_{m-1})$ and adds the results to SFS_n. (3) The PCV extractors compute the PCV from the SFS_i in each dimension using BISCAL. (4) The CV extractors calculate the CVs by combining the PCVs obtained in the previous step using BISCAL. (5) The $TR^{n+1}(f_i, f_j)$ for all the filter pairs in the TFP are detected using BISCAL in the TR extractors. (6) The conflict detector and the classifier classifies the conflicts into errors, warnings, and no error and warning filters according to the $TR^{n+1}(f_i, f_j)$ for all the filter pairs in the TFP.

The computation of TR^{n+1} of the filters are completely the same as the extraction of TR^n of the filters discussed in chapter 4 except the computation in the time divisor. As we have discussed the computational steps in extracting SFS_n through temporal analysis in chapter 5, we omit the computational steps in this chapter.

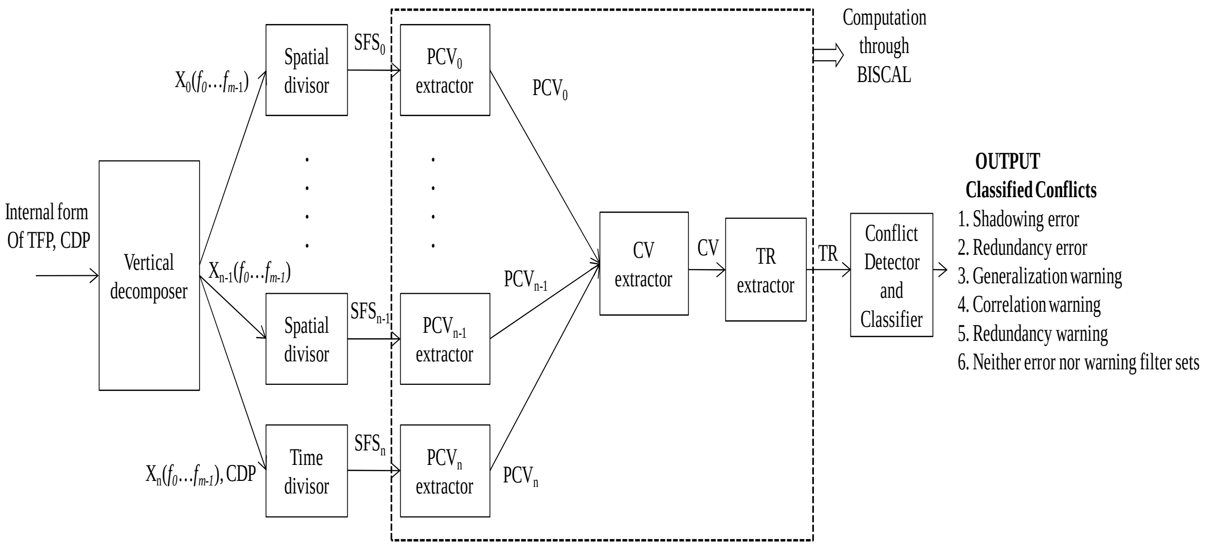


Figure 6.1 Overview of the conflict detection system architecture through simultaneous analysis

6.2 Iterative Analysis

The system overview of this architecture is shown in Figure 3.2. The system receives the internal form of the TFP and CDP as the input and follows the procedure given below. (1) The time divisor decomposes the X_i ($f_0 \ldots f_{m-1}$) into ‘p’ divisions either through geometry-based temporal analysis or the topology-based temporal analysis. (2) For each interval from T_0 to T_{p-1}, each filter set from SFS_n which has multiple filters, are extracted and given to the spatial divisor for n-dimensional spatial analysis. (3) Project the filters on the n-dimensional space and decomposes them in their boundaries to form decomposed filter sets. (4) The TR^n of the filters are computed from the decomposed filter sets through BISCAL as we have previously discussed in chapter 4. (5) The results of the TR^n of the filters are gathered and the $TR^{n+1}(f_i, f_j)$ of the TFP is computed through BISCAL. (6) The conflict detector and classifier detects the conflicts and classifies the conflicts into errors, warnings, and no error and warning filters according to the $TR^{n+1}(f_i, f_j)$ for all the filter pairs in the TFP.

6.3 Evaluations

6.3.1 Experiments

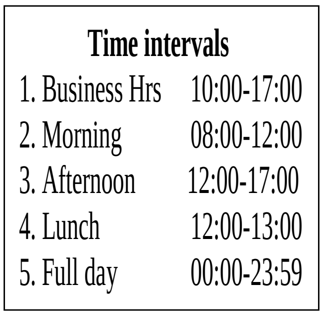
We have developed the prototype systems in JAVA programming language and the experiments were performed on Intel (R) Core (TM) i5 CPU 750 @ 2.67 GHz 2.67 GHz with 4.00GM RAM running on Windows 7 professional. Due to the unavailability of publicly used TFPs, we have synthesized a number of policies, TFP (m) which was synthesized from FP (m), which has *m* filters as follows:

(1) FP (m)

We have synthesized a firewall policy without time-field, FP(m), by using a firewall policy of 100 filters without the time-field, FP_L, which has five fields, SrcIP, DesIP, SrcPort, DesPort and Pro and is used in our laboratory, as follows: it is a first-m filters of the FP_L in case $m \leq 100$ while it is a combination of the FP_L and new filters which are synthesized by using a randomly selected filter from FP_L in case m>100.

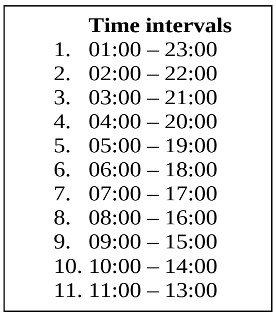
(2) TFP (m)

We have converted the FP (m) to TFP (m) by adding the values of TIME and DAY in the ActTIME field as follows: the kth filter in TFP (m) is a combination of the kth filter of FP (m) and generated values of TIME as given by the two time charts shown in figure 6.2. The time chart shown in Figure 6.2 (a) is created based on control the network traffic in time in industrial environments. The time chart shown in figure 6.2(b) is created to create large number of conflicts in the TFPs to compare the system performances.



Time intervals	
1. Business Hrs	10:00-17:00
2. Morning	08:00-12:00
3. Afternoon	12:00-17:00
4. Lunch	12:00-13:00
5. Full day	00:00-23:59

(a) Case I



Time intervals
1. 01:00 – 23:00
2. 02:00 – 22:00
3. 03:00 – 21:00
4. 04:00 – 20:00
5. 05:00 – 19:00
6. 06:00 – 18:00
7. 07:00 – 17:00
8. 08:00 – 16:00
9. 09:00 – 15:00
10. 10:00 – 14:00
11. 11:00 – 13:00

(b) Case II

Figure 6.2 Time Charts

We randomly generated values of DAY by adding a single non-periodic filter which is active on 12 Sept 2012 and the remaining *m-1* DAY values are selected as periodic filters. The remaining *m-1* filters are periodic filters and it is controlled by two parameters, p1-the percentage of FE and p2- the percentage of FW. We varied the percentage of p1 and p2 by maintaining the sum of p1 and p2 as 1. For example, if we add 20% of p1 filters, then we add 80% of p2 filters to make the sum of p1 and p2 as 1.

As we know that, there are two architectures to detect the conflicts of TFP and two types of temporal analysis, we construct four systems by making different combinations of the systems and the type of temporal analysis. We found that there are four combinations of systems to detect the conflicts in TFP as follows,

1. **SystemA**: A conflict detection system through Iterative Analysis which incorporates geometry-based temporal analysis.

2. **SystemB**: A conflict detection system through Iterative Analysis which incorporates topology-based temporal analysis.
3. **SystemC**: A conflict detection system through Simultaneous Analysis which incorporates the geometry-based temporal analysis.
4. **System D**: A conflict detection system through Simultaneous Analysis which incorporates the topology-based temporal analysis.

We compare the above four systems and finds the system which performs well in terms of computation time.

Experiment I: We evaluated the usefulness of the proposed system by comparing the number of conflicts detected with and without considering the time field and by investigating the percentage of false positive results.

Experiment II: We evaluated the feasibility of the proposed system by investigating the computation time of the four systems in terms of two parameters: the number of filters and the length of CDP.

6.3.1 Results of Experiment I

We have compared the ratio of conflicts, which the two systems detected in the TFP (100). We have performed this experiment to determine the percentage of false positive results that the prevailing techniques can generate during conflict detection for TFPs. We introduce a class of conflicts c, which shows an error from among *{shadowing error, redundancy error, correlation warning, generalization warning, redundancy warning and no error or warning}*, which have been classified in chapter 2, and define the ratio of each class of conflicts as follows:

$$R(c) = \frac{\text{Number of class-c conflicts by considering time}}{\text{Number of class-c conflicts without considering time}}$$

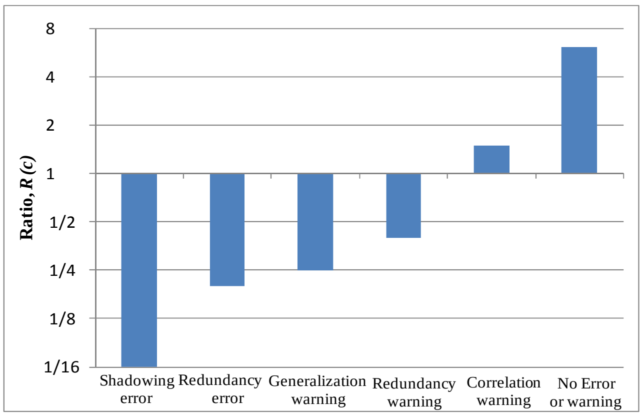


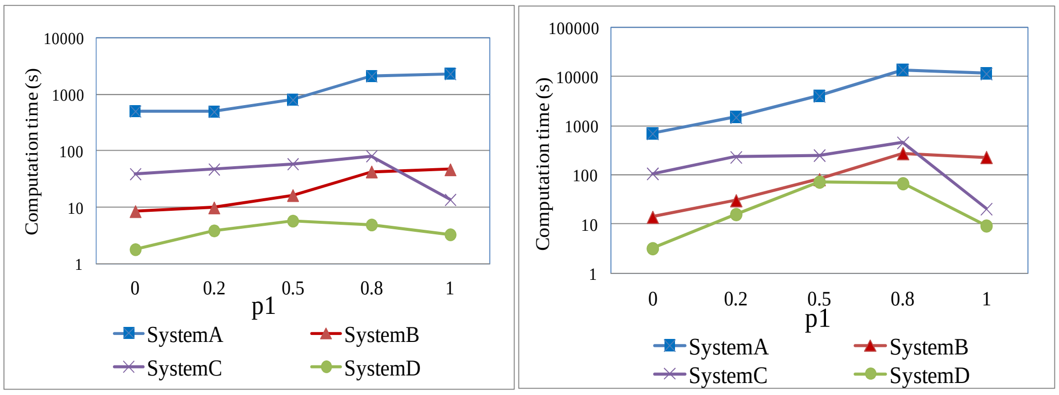
Figure 6.3 The ratio of conflicts with and without considering time

The figure 6.3 shows the relationship between the class of conflicts and R. In this Figure, we find that the ratio of *shadowing error* is 1/16 and that of *redundancy error* is 1/5. We also find that the filters exhibiting *generalization warning* and *redundancy warning* also considerably reduced, as shown in Figure 6.3. The increase in the *correlation warning* shows that the fully covered filters become partially covered in the (n + 1)-dimensional space and a large increase in the *no error and warning filter* shows that nearly 50% of the filters become disjoint in the temporal space. *Since the workload of the administrator is directly proportional to the effort towards reconfiguration of the conflicting filters, the workload of the administrator is reduced drastically as the number of conflicting filters is reduced.* Therefore, we conclude that the proposed system takes eliminate false positive results completely and thereby, help to reduce the workload of the network administrator to a reasonable level.

6.3.2 Results of Experiment II

The SystemA and SystemC compute the CDP through method1 as [13 Sep 2011, 12 Sep 2012] and the conflict detection period is computed through periodic cycle treatment for SystemB and SystemD. We have conducted experiments with four different systems and have found the computation time to find the conflicts in TFP and plotted the graph where p1 is plotted in X-axis and computation time is plotted in Y-axis. The graphs shown in figure 6.4 (a) and (b) shows the computation time of the system when the time chart given by case I and Case II are experimented respectively.

We can infer that from the graphs that, when the ratio of p1 is between 0.5 and 0.8, computation time is comparatively higher due to the chances of the filters getting conflicted is large and resulting in large number of conflicting filter sets. When the ratio of p1 is low, computation time is lower due to the lower number of conflicting filter sets. When the ratio of p1 is high, computation time gradually decreases because of the lower number of conflicting intervals even though the number of conflicting filters is large. When we analyze the graphs, we found that, SystemC outperforms SystemB when p1 is 1 and SystemB outperforms SystemC when p1 is between 0 and 0.8. Among all the values, SystemD performs better than all the systems because of the advantages of the mapping in the shortest CDP and finding the *n+1*-dimensional TR of the filters through simultaneous approach rather than interval basis. Therefore **SystemD** could able to reduce the computation time to a minimum level as the repetition of filter sets are completely removed and also the irrelevant data are removed in the initial stage of computation.



(a) Case I (b) Case II

Figure 6.4 Comparative Analysis of the different systems

6.4 Conclusion

We have described the implementation of the two architectures to compute the TR^{n+1} of the filters to detect conflicts in TFPs. By our proposed system, the false positive results can be completely avoided by computing the exact conflicts in the TFPs. We have also found that SystemD performs better than the other combinations of the architectures. As the number of conflicting filters is drastically reduced in TFPs, the workload of the administrator is also reduced and therefore our proposed systems would be very helpful to the network administrator in making decisions to discard the conflicts in the firewall policies.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Firewall security, like any other technology, requires proper management to provide proper security service. Thus, just having a firewall on the boundary of a network may not necessarily make the network any secure. One reason of this is the complexity of managing firewalls filters and the potential network vulnerability due to filter errors and warnings (e.g. shadowing, correlation) called conflicts. Conflict detection through spatial analysis has lot of advantages like detection of conflicts from the combinations of filters, but the drawback is that it takes large computation time and memory to detect the conflicts in the firewall policies. Even though there are lots of conflict detection techniques, they cannot be applied to detect conflicts in time-based firewall policies as the time-field is ignored during conflict detection. This paper provides a new methodology of detecting conflicts in the time-based firewall policies through two architectures. We have three main contributions for the conflict detection in TFPs. We have developed a topology-based spatial analysis through BISCAL which discards the unnecessary computations and could able to discard the drawbacks of the previous approaches based on geometry. We have developed methods for temporal analysis and have proposed two system architectures to compute the conflicts in TFPs. We have made experiments with the proposed systems through different architectures and have validated the better performance systems. Our contributions for the conflict detection system for TFPs will be a useful tool for the network administrator in the management of firewall policies to reconfigure the firewall policies efficiently.

7.2 Future Work

Our future work focuses on the detection of conflicts caused by the combination of filters through the topology-based spatial analysis through BISCAL in TFPs. Yin *et al* have developed a conflict detection system caused by combinations of filters through geometry-based spatial analysis for FPs [4]. We will extend our research in finding the conflicts from the combinations of filters through BISCAL for TFPs. As we have already proved in our research that topology performs better than geometry, we will develop a topology-based conflict detection system for the conflicts caused by combinations of filters for TFPs. Further we would like to extend our research to make a visualization tool for an easy understanding of the different types of conflicts to the network administrator. For example, if there is large number of conflicts, it is difficult to find which filter needs to be reconfigured first. Therefore, we will develop a visualization tool for the network administrator which shows a sequence to reconfigure the filters in the policies.

Publications

Journals

[1] Thanasegaran Subana, Yuichiro Tateiwa, Yoshiaki Katayama, Naohisa Takahashi, "Design and Implementation of Conflict Detection System for Time-based Firewall Policies," JNIT: International Journal of Next Generation Information Technology, Vol. 2, No. 4, pp. 24-39, Nov 2011.

[2] Thanasegaran Subana, Yi Yin, Yuichiro Tateiwa, Yoshiaki Katayama, Naohisa Takahashi, "A Topology-Based Conflict Detection System for Firewall Policies using Bit-Vector-Based Spatial Calculus," IJCNS: International Journal of Communication Network and System Sciences, Vol.4, No:11, pp.683-695, Nov 2011.

International Conferences

[1] Thanasegaran Subana, Yi Yin, Yuichiro Tateiwa, Yoshiaki Katayama, Naohisa Takahashi: "A Topological Approach to Detect the Conflicts in the Firewall Policies," In Proc. of 23rd IEEE International Conference on IPDPS 2009, SSN-1569173665-paper-3.pdf, Rome, Italy, May 2009. **(IPSJ's Tokai-section Student Best Paper Award).**

[2] Thanasegaran Subana, Yuichiro Tateiwa, Yoshiaki Katayama, Naohisa Takahashi, "Simultaneous Analysis of Time and Space for Conflict Detection in Time-based Firewall Policies," In Proc. of 10th IEEE International Conference on CIT, pp.1015-1021, Bradford, UK, June 2010.

[3] Thanasegaran Subana, Yuichiro Tateiwa, Yoshiaki Katayama, Naohisa Takahashi, "An improved conflict detection system with periodic cycle treatment for time-based firewall policies," In Proc. of 19th IEEE ICCCN 2010, pp.1-8, Zurich, Switzerland, Aug 2010.

Technical Reports

Thanasegaran Subana, Yi Yin, Yuichiro Tateiwa, Yoshiaki Katayama, Naohisa Takahashi, "BISCAL: Bit Vector Based Spatial Calculus for Analyzing the Misconfigurations in Firewall," In Proc. of IEICE IA Technical Report, 108 (409), pp.101-106, Tokyo, Japan, Jan 2009. **(Student Encouragement Award)**

Domestic Conferences

[1] Thanasegaran Subana, Yi Yin, Yoshiaki Katayama, Naohisa Takahashi, “BISCAL: Bit Vector Based Spatial Calculus for Analyzing Spatial Relationships between Filters,” In Proc. of Tokai Rengo conference, 0079, Aichi Prefectural University, Japan, Sep 2008. [In Japanese]

[2] Thanasegaran Subana, Yuichiro Tateiwa, Yoshiaki Katayama, Naohisa Takahashi, "Detection of Conflicts in Time-dependent Firewall Policies", In Proc. of IEICE General Conference, 187, Matsuyama, Japan, March 2009.

Scholarships and Grants

Monbukagakusho Scholarship, from October 2006 to March 2012.

Research grant from the Telecommunications Advancement Foundation, August 2010.

Research grant from the Hori Information Science Promotion Foundation, from April 2011 to March 2012.

Bibliography

[1] Max J. Egenhofer, “A formal definition of binary topological relationships,” LNCS 367/1989, pp.457-472, USA, 1989.

[2] A. Wool, “A quantitative study of firewall configuration errors,” Computer, vol.37, pp.62-67, 2004.

[3] N. Takahashi, “A systolic sieve array for real-time packet classification,” IPSJ Journal, vol.42, no.2, pp.146-166, 2001.

[4] Y. Yin, Y. Katayama, N. Takahashi, “Detection of conflicts caused by a combinations of filters based on spatial relationships,” In IPSJ Journal, vol.49, pp.3121-3135, Sept. 2008.

[5] T. Subana, Y.Yin, Y. Tateiwa, Y. Katayama, N. Takahashi, "BISCAL: Bit Vector Based Spatial Calculus for analyzing the misconfigurations in firewall," Proc of IEICE IA Technical Report, 108 (409), pp.101-106, Tokyo, Japan, Jan 2009.

[6] T. Subana, Y.Yin, Y. Tateiwa, Y. Katayama, N. Takahashi, "A topological approach to detect the conflicts in the firewall policies," Proc. of 23rd IEEE International Conference on IPDPS 2009, SSN-1569173665-paper-3.pdf, Rome, Italy, May 2009.

[7] T. Subana, Y. Tateiwa, Y. Katayama, N. Takahashi, "Simultaneous analysis of time and space for conflict detection in time-based firewall policies," Proc. of 10th IEEE International Conference on CIT, pp.1015-1021, Bradford, UK, June 2010.

[8] T. Subana, Y. Tateiwa, Y. Katayama, N. Takahashi, "An improved conflict detection system with periodic cycle treatment for time-based firewall policies," Proc. of 19th IEEE ICCCN 2010, pp.1-8, Zurich, Switzerland, Aug 2010.

[9] D. Eppstein, S. Muthukrishnan, "Internet packet filter management and rectangle geometry," Proc. Of 12th Annual ACM-SIAM SYM, Washington, pp.827-835, USA, 2001.

[10] H. Hamed, E. Al-Shaer, "Taxonomy of conflicts in network security policies," IEEE Communication Magazine, vol.44, no.3, pp.134-141, 2006.

[11] E. Al-Shaer and H.Hamed, "Modeling and management of firewall policies", In IEEE Transactions on Network and Service Management, Vol.1-1, Apr.2004.

[12] E. Al-Shaer, H.Hamed, R.Boutable, and M.Hasan, "Conflict classification and analysis of distributed firewall policies", In IEEE Journal on Selected Areas in Communication, 23(10): pp.2069-2084, 2005.

[13] E. Al-Share and H.Hamed, "Firewall policy Advisor for anomaly detection and rule editing", In Proc. Of 8th IEEEE/IFIP Integrated Management, (IM'2003), March 2003.

[14] E. Al-Share and H.Hamed, "Design and implementation of firewall policy advisor tools", Depaul CTI Technical Report, CTI-TR-02-006, August 2002.

[15] K. Golnabi, R.K. Min, L. Khan, E. Al-Shaer, "Analysis of firewall policy filters using data mining techniques," Proc. Of IEEE NOMS 2006, pp.305-315, Canada, April 2006.

[16] L. Yuan, J. Mai, Z. Su, H. Chen, P. Mohapatra, "FIREMAN: a toolkit for firewall modeling and analysis," Proc. Of IEEE Symposium on Security and Privacy, pp.199-213, Oakland, May 2006.

[17] A. Mayer, A. Wool, E. Ziskind, "FANG: a firewall analysis engine," Proc. of IEEE Symposium on Security and Privacy, pp.177-187, Oakland, May 2000.

[18] A. Wool, "Architecting the lumeta firewall analyzer," Proc. Of 10th Conf. USENIX Security Symposium, pp.7-7, USA, Aug 2001.

[19] B. Zhang, E. Al-Shaer, R. Jagadeesan, J. Riely, C. Pitcher, "Specifications of a high-level conflict-free firewall policy language for multi-domain networks," Proc. Of SACMAT'07, pp.185-194, Sophia Antipolis, France, June 2007.

[20] A. Hari, S. Suri, G. Parulkar, "Detecting and resolving packet filter conflicts," Proc. of IEEE INFOCOM 2000, pp.1203-1212, Israel, Mar. 2000.

[21] V. Capretta, B. Stepien, A. Felty, S. Matwin, "Formal correctness of conflict detection for firewalls," Proc. of ACM workshop on Formal Methods in Security Engineering, Virginia, pp.22-30, USA, Nov 2007.

[22] A. Liu, E. Torng, C. Meiners, "Firewall compressor: An algorithm for minimizing firewall policies," Proc. of 27th IEEE INFOCOM, Phoenix, Arizona, pp. 176-180, USA, April 2008.

[23] M. Yoon, S. Chen, Z. Zhang, "Minimizing the maximum firewall rule set in a network with multiple firewalls," IEEE Transactions on Computers, vol.59, no.2, pp.218-230, Feb 2010.

[24] G. Misherghi, L. Yuan, Z. Su, C.-N. Chuah, H. Chen, "A general, framework for benchmarking firewall optimization techniques," IEEE Transactions on Network and Service Management, vol.5, no.4, pp.227-238, Dec. 2008.

[25] H.G. Verizon, K.A. Ahmat, "Fast and scalable method for resolving anomalies in firewall policies," Proc. Of 14th IEEE Global Internet Symposium 2011 at IEEE INFOCOM 2011, pp.839-844, April 2011.

[26] H. Hu, G.J. Ahn, K. Kulkarni, "FAME: A firewall anomaly management environment," Proc. Of ACM SafeConfig'10, ISBN: 978-1-4503-0093-3, Oct. 2010.

[27] T. Srinivasan, N. Dhanasekar, M. Nivedita, R. Dhivyakrishnan, A.A. Azeezunnisa, "Scalable and parallel aggregated bitvector packet classification using prefix computation model," Proc. Of Int SYM on PAR ELEC 2006, pp.139-144, Bialystok, 2006.

[28] T.V. Lakshman, "High-speed policy based packet forwarding using efficient multi-dimensional range matching," Proc. Of ACM SIGCOMM 98, vol.28, pp.203-214, Vancouver, Sep 1998.

[29] S. Singh, F. Baboescu, G. Varghese, J. Wang, “Packet classification using multidimensional cutting,” Proc. of ACM SIGCOMM 03’, pp.213-224, Germany, Feb 2003.

[30] K. Matsuda, “A packet filtering filters compression by decomposing into matrixes,” IPSJ Journal, vol.48, no.10, pp.3357-3364, 2007 [in Japanese].

[31] Yi Yin, Kazuaki Hida, Yoshiaki Katayama, Naohisa Takahashi, "Implementation of filter reverse search system based on spatial relationships of filters", JCIT: Journal of Convergence Information Technology, Vol. 3, No. 2, pp. pp.6 - pp.12, 2008.

[32] Young-Long Chen, Ying-Chen Chen, "Dynamic managements of the firewall policy to mitigate DDoS attacks", JCIT: Journal of Convergence Information Technology, Vol. 6, No. 8, pp. 292-298, 2011.

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

[34] https://www.cisco.com/en/US/docs/security/pix/pix63/release/notes/pixrn634.html

[35] http://www.pcis.com/products/astaro_firewall.html

[36] http://linux.die.net/man/8/iptables