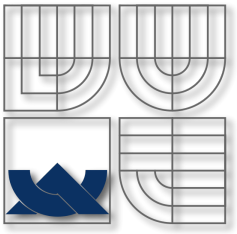
BRNO UNIVERSITY OF TECHNOLOGY



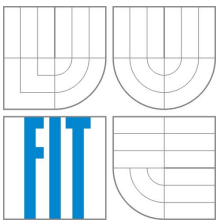
Network-wide Security Analysis

by

Hidda Marakkala Gayan Ruchika de Silva

A dissertation submitted in partial fulfillment for the
degree of Doctor of Philosophy
(Computer Science and Engineering)

Faculty of Information Technology
Department of Information Systems



Supervised by:
Prof. Miroslav Švéda and Dr. Ondřej Ryšavý

October 2011

Declaration of Authorship

I, Hidda Marakkala Gayan Ruchika de Silva, declare that this thesis titled, 'Network-wide Security Analysis' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University (since October 2008).
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

“ PhD is a tremendous distance swum overcoming tides, storms, and vociferous fish along with physical and mental ailments, in an arduous attempt of conquering the ocean and the world of knowledge and skills.”

BRNO UNIVERSITY OF TECHNOLOGY

Abstract

Faculty of Information Technology
Department of Information Systems

Doctor of Philosophy

by Hidda Marakkala Gayan Ruchika de Silva

The objective of the research is to model and analyze the effects of dynamic routing protocols. The thesis addresses the analysis of service reachability, configurations, routing and security filters on dynamic networks in the event of device or link failures.

The research contains two main sections, namely, modeling and analysis. First section consists of modeling of network topology, protocol behaviors, device configurations and filters. In the modeling, graph algorithms, routing redistribution theory, relational algebra and temporal logics were used. For the analysis of reachability, a modified topology table was introduced. This is a unique centralized table for a given network and invariant for network states. For the analysis of configurations, a constraint-based analysis was developed by using XSD Prolog. Routing and redistribution were analyzed by using routing information bases and for analyzing the filtering rules, a SAT-based decision procedure was incorporated. A part of the analysis was integrated to a simulation tool at $OMNeT^{++}$ environment.

There are several innovations introduced in this thesis. Filtering network graph, modified topology table, general state to reduce the state space, modeling devices as filtering nodes and constraint-based analysis are the key innovations. Abstract network graph, forwarding device model and redistribution with routing information are extensions of the existing research. Finally, it can be concluded that this thesis discusses novel approaches, modeling methods and analysis techniques in the area of dynamic networks. Integration of these methods into a simulation tool will be a very demanding product for the network designers and the administrators.

Acknowledgements

It is a pleasure to proudly announce aloud that I was capable of presenting this thesis successfully due to the invaluable contributions offered in many ways by the several recognized academics at Brno University of Technology that need be mentioned with gratitude. Firstly, I am immensely grateful to my advisor, Prof. Miroslav Švéda, for giving me this golden opportunity in conducting my PhD studies under his supervision on networked and embedded systems research group with all the support from the inception to the final stage. My PhD research and this thesis would not have been possible without the greatest support, directions and the subject knowledge given by my co-supervisor, Dr. Ondřej Ryšavý, for which I owe a debt of gratitude. I offer a huge thank you to Dr. Petr Matoušek for all the assistance given on enrolling me to this program, aligning me to the research and the noteworthy guidance offered.

I am ever thankful to Dean, Dr. Jaroslav Zendulka, and Vice Dean, Prof. Tomáš Hruška, for admitting me to the PhD studies in this university. Also, I would like to express my indebtedness and gratitude to Brno University of Technology and the staff for giving me unvarying courtesy and facilities on all the academic, financial, technical and library sectors to successfully make my studies possible. Further, I am grateful for the projects TeamIT, Security-Oriented Research in Information Technology, and Safety and Security of Networked Embedded System Applications, for providing me the financial support during my study.

This thesis would not have been attainable, if not for all the encouragement and love given freely at all times by my father, Mr. Nanda de Silva, my mother, Mrs. Nalini de Silva, and my two sisters. I take this opportunity to thank my relatives and friends who have supported me in numerous ways to conduct my studies.

Finally, I would like to show my deepest gratitude for the lovely country Czech Republic, which enabled my dream come true.

Contents

Declaration of Authorship i

Abstract iii

Acknowledgements iv

List of Figures viii

List of Tables x

Abbreviations xi

Symbols xiii

1 Introduction 1
- 1.1 Overview . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Motivation . . . 3
- 1.4 Significance and Objectives . . . 4
  - 1.4.1 Modeling . . . 4
  - 1.4.2 Analysis . . . 5
- 1.5 Proposed Research Methodology and Plan . . . 6
  - 1.5.1 Modeling . . . 6
  - 1.5.2 Filtering and Routing . . . 7
  - 1.5.3 Configuration Abstraction . . . 8
  - 1.5.4 Analysis . . . 8
- 1.6 Contribution . . . 9
- 1.7 Structure of the Thesis . . . 12

2 State-of-the Art 14
- 2.1 Network Reachability . . . 15
- 2.2 Verification of Security Policies . . . 21

3 Modeling Networks 25
- 3.1 Abstract Network Model . . . 27

3.2 Filtering Network Model . . . 29
3.2.1 Model Transformation . . . 34
3.3 Graph Transformation . . . 34
3.3.1 Graph Transformation Definitions . . . 35
3.3.2 Graph Grammar . . . 36
3.4 Model of Forwarding Devices and Routing Information . . . 42
3.4.1 Modeling Forwarding Devices . . . 43
3.4.2 Cost Function for RIP, OSPF and EIGRP . . . 44
3.4.3 Representing Routing Information . . . 45
3.4.4 Forwarding State . . . 45
3.5 Discussion . . . 47

4 Reachability Analysis **49**
4.1 Modified Topology Table . . . 50
4.1.1 Computing General State . . . 51
4.2 State Loss Graph Patterns . . . 56
4.2.1 Identify and Eliminate State Loss . . . 57
4.2.2 Computation of GS on State Loss Pattern . . . 60
4.2.3 Higher Order State Loss Patterns . . . 62
4.3 Reachability Analysis . . . 63
4.4 MTT Extension for OSPF Configured Networks . . . 65
4.4.1 OSPF Overview . . . 66
4.4.2 OSPF Modeling with MTT . . . 67
4.5 MTT with Combination of Routing Protocols . . . 69
4.6 Results and Discussion . . . 70

5 Constraint-based Analysis **74**
5.1 Constraint Data Model . . . 75
5.2 Configuration Analysis . . . 80
5.2.1 Access Control Lists . . . 80
5.2.2 Network Address Translation . . . 84
5.2.3 Type of Service Marking . . . 92
5.3 Constraint Queries . . . 93
5.3.1 Tunnel Configuration Validation . . . 94
5.3.2 Waypoints and Forbidden Paths . . . 95
5.3.3 Rate Limitations . . . 96
5.4 Results and Discussion . . . 96

6 Routing Analysis **99**
6.1 Routing Protocols . . . 100
6.1.1 Routing Approximations . . . 101
6.1.2 Directly Connected Networks . . . 102
6.1.3 Calculation of Static RIBs . . . 102
6.1.4 Calculation of Dynamic RIBs . . . 103
6.2 Distance Vector Protocols . . . 104
6.2.1 Filtering Routing Updates . . . 105
6.2.2 Analyzing the Effects of Routing Filters . . . 106

6.3 Redistribution . . . 108
6.3.1 Redistribution Between Multiple IGP Instances . . . 108
6.3.2 Redistribution Between Multiple IGP Networks . . . 112
6.4 Results and Discussion . . . 115

7 Firewall Policy Analysis 117
7.1 Representation of Packet Filters . . . 118
7.1.1 List-based Packet Filters . . . 118
7.1.2 Resolving Conflicts . . . 119
7.1.3 Rule Sets . . . 119
7.2 Filter Representation . . . 120
7.2.1 Address Scheme . . . 121
7.2.2 Representation of Rule Sets . . . 122
7.2.3 Verification of Security Policies . . . 122
7.3 Verification Method . . . 123
7.3.1 Verification of a Single Filter . . . 123
7.3.2 Verification of Cascaded Filters . . . 128
7.4 Network-wide Analysis . . . 129
7.4.1 Encoding as SMT Problem . . . 129
7.4.2 Most-General Counter Example . . . 130
7.5 Results and Discussion . . . 131

8 Analysis with Simulation tools 133
8.1 Building Models upon Configuration Files . . . 135
8.2 Detecting Critical Elements . . . 136
8.2.1 Formal Model of the Network . . . 136
8.2.2 Reachability Analysis . . . 137
8.3 Analyzing Convergence Delays Using OMNeT++ . . . 140
8.3.1 Setting Simulation Environment . . . 141
8.3.2 Simulation Scenarios . . . 141
8.4 Results and Discussion . . . 142

9 Conclusions and Suggestions for Future Work 144

A Rubin's Algorithm 150

B Random and Ring Networks 151

C Ordering Networks 152

D Floyd-Warshall Algorithm 153

E An Example of a Prolog Query 154

Bibliography 155

List of Figures

1.1 Scope of the research . . . 6
1.2 Structure of the Thesis . . . 12

3.1 Structure of Chapter 3 . . . 26
3.2 Types of network graphs . . . 27
3.3 A device node model . . . 30
3.4 Converting loopback paths . . . 32
3.5 Point-to-multipoint networks . . . 33
3.6 Graph transformation . . . 37
3.7 Expansion rule, $p_1 = (L_1 \overset{l_1}{\leftarrow} K_1 \overset{r_1}{\rightarrow} R_1)$ . . . 38
3.8 Full-Mesh Rule, $p_2 = (L_2 \overset{l_2}{\leftarrow} K_2 \overset{r_2}{\rightarrow} R_2)$ . . . 39
3.9 Termination rule, $p_3 = (L_3 \overset{l_3}{\leftarrow} K_3 \overset{r_3}{\rightarrow} R_3)$ . . . 39
3.10 An example of GT Part-1 . . . 40
3.11 An example of GT Part-2 . . . 41
3.12 An example of router model . . . 43
3.13 An example of network topology . . . 46

4.1 Structure of Chapter 4 . . . 50
4.2 Features of Modified Topology Table . . . 51
4.3 Mapping between different states . . . 52
4.4 Z–Function . . . 53
4.5 A dynamic network . . . 54
4.6 Construction of state loss pattern . . . 58
4.7 Parallel edges . . . 59
4.8 State loss pattern . . . 60
4.9 Adding vertical branches to the state loss pattern . . . 62
4.10 Adding horizontal branches to the state loss pattern . . . 62
4.11 Typical OSPF configured network . . . 66
4.12 MTT computation process for OSPF . . . 68

5.1 Structure of Chapter 5 . . . 74
5.2 An example network topology . . . 76
5.3 A sequence of events in NAT session lifetime . . . 85

6.1 Structure of Chapter 6 . . . 100
6.2 An example of static network configuration and network RIBs . . . 103
6.3 An example of route filtering and computation of network RIBs . . . 106
6.4 Network topology with RIP and OSPF . . . 110

6.5 Routing redistribution and route selection 111
6.6 Example of two IGP networks 114

7.1 Structure of Chapter 7 118
7.2 Example network of an organization 129

8.1 Structure of Chapter 8 134
8.2 Structure of the analysis process 134
8.3 CESNET Network 137
8.4 CESNET network with critical links 140

B.1 Ring network with degree four 151
B.2 Random network with degree three 151

C.1 Large and small world networks 152
C.2 Effects of network ordering 152

List of Tables

3.1 Filter map . . . 46
3.2 NFIB for state q and r . . . 47

4.1 Modified Topology Table . . . 56
4.2 Default administrative distances . . . 70
4.3 Time consumption for building MTT . . . 71

5.1 Flow constraint table . . . 75
5.2 An example of flow constraint table . . . 76
5.3 Transformation constraint table . . . 78
5.4 An example of filter constraint table . . . 80
5.5 An example of static NAT constraint table . . . 86
5.6 An example of dynamic NAT constraint table . . . 87
5.7 An example of NAPT constraint table . . . 90

6.1 MTT for IGP network-1 . . . 114
6.2 MTT for IGP network-2 . . . 114

7.1 Network field selectors . . . 120

8.1 Cost function for CESNET network . . . 138
8.2 Costs and path π^k between R_{23} and R_1 . . . 139
8.3 Scenario definitions . . . 142
8.4 Measured values . . . 142

Abbreviations

ABR	**A**rea **B**order **R**outers
ACL	**A**ccess **C**ontrol **L**ist
ANG	**A**bstract **N**etwork **G**raph
ANSA	**A**utomated **N**etwork-wide **S**ecurity **A**nalysis
ANTLR	**AN**other **T**ool for **L**anguage **R**ecognition
ASBR	**A**utonomous **S**ystem **B**order **R**outers
BGP	**B**order **G**ateway **P**rotocol
BUT	**B**rno **U**niversity of **T**echnology
CP	**C**ritical **P**oints
CVE	**C**ommon **V**ulnerabilities and **E**xposures
DPO	**D**ouble **P**ush **O**uts
FDD	**F**irewall **D**ecision **D**iagram
FIB	**F**orwarding **I**nformation **B**ase
FIT	**F**aculty of **I**nformation **T**echnology
FML	**F**low-based **M**anagement **L**anguage
FNG	**F**iltering **N**etwork **G**raph
GG	**G**raph **G**rammar
GP	**G**raph **P**roduction
GRE	**G**eneric **R**outing **E**ncapsulation
GS	**G**eneral **S**tate
GT	**G**raph **T**ransformation
GTS	**G**raph **T**ransformation **S**ystem
HSRP	**H**ot **S**tandby **R**outer **P**rotocol
IDD	**I**nterval **D**ecision **D**iagram
IGP	**I**nterior **G**ateway **P**rotocol

IOS	**I**nter-network **O**perating **S**ystem
LSA	**L**ink **S**tate **A**dvertisement
MPLS	**M**ulti**P**rotocol **L**abel **S**witching
MTT	**M**odified **T**opology **T**able
MTU	**M**aximum **T**ransmission **U**nit
NAPT	**N**etwork **A**ddress **P**ort **T**ranslation
NAT	**N**etwork **A**ddress **T**ranslation
NSSA	**N**ot **S**o **S**tubby **N**etwork
OSI	**O**pen **S**ystem **I**nterconnection
OSPF	**O**pen **S**hortest **P**ath **F**irst
RIB	**R**outing **I**nformation **B**ase
RIP	**R**outing **I**nformation **P**rotocol
SFG	**S**ervice **F**low **G**raph
SPSL	**S**ecurity **P**olicy **S**pecification **L**anguage
SPT	**S**hortest **P**ath **T**ree
ToS	**T**ype **o**f **S**ervice
TG	**T**yped **G**raph
UP	**U**niversal **P**oints

Symbols

symbol	name
L	set of all links
l	a link
R	set of all devices
C	cost function
F	set of filtering functions
r	a device
π	a path
G, H	graphs
N_A	abstract network graph
N_F	filtering network graph
E	set of edges
V	set of vertices
N	network
$\mathcal{N}$	natural numbers
$\mathcal{T}$	graph transformation function
M	link bit vector matrix
$M_{max(a,b)}$	biggest link bit vector matrix for path a and b
Π	all paths between a given source and a destination
$[\Pi]_C$	paths are in the oder of Cost (C)
$MTT(s, d)$	selected paths in MTT between source s and destination d
S_G^k	general state for k^{th} path π^k
S^k	network state for k^{th} path π^k

This thesis is dedicated to my parents

for their love, endless support
and encouragement.

Chapter 1

Introduction

1.1 Overview

Computers have become an essential component in most of critical operations. These critical operations are distributed and accessible from different locations. Use of the applications and the services over computer networks are rapidly increasing day by day and computer users are becoming more advanced and demanding new services, quality of services, and high availability. These demands require reliable computer networks. While enabling the reliable services to users, we need to avoid unauthorized access by implementing proper filtering rules. Current network communication has moved from a simple data exchange to integration of different services such as transmitting voice and video. To achieve these user demands, we must have proper analysis methods to identify design problems, evaluate the reliability and security of the services in different network conditions such as device or link failures, configuration errors and heavy loads.

Network specialists are expected to fulfill customer requirements, while considering the limits of underlined technologies. Therefore, the network design has become a complex and challenging task. The goal of a proper network design is to provide reliable network services under acceptable network protection. Network design includes planning of network topology, network devices, IP addressing and routing, security measures, reachability and quality of network services, and applications on higher levels. Once the design phase is finished, the deployment phase is launched. It consists of installation, which includes physical interconnections of devices, setting up their configurations, and finally network troubleshooting, in order to assure network functionality. Identification of potential problems in early design phase is a challenging task. This identification requires extra techniques and methodologies that verify and validate the correctness of the design process.

Implementation of routing to deliver the required services over a network is one of the key tasks in network design. Configuration of proper routing enables the reachability of required services. This can be easily evaluated at stable network state by using various testing tools. To enable uninterrupted services to users, network designers have moved from configuration of conventional static routing to configuration of advanced dynamic routing protocols. Routing design with dynamic routing protocols is a very complex task, in practice, as contemporary converged networks have to achieve many different objectives such as basic reachability, network security, quality of services required by converged applications, resilience, and scalability. These protocols contain many implementation options to meet the design goals, where each of them imposes certain constraints and side effects. To validate routing designs, complex networks are usually set up and troubleshoot in a laboratory environment to find the acceptable design and to correct possible configuration errors.

Once the services are available for users, it is important to restrict access for the unwanted parties and protect the services. This is mainly achieved by placing a firewall or an implementation of firewall rules on routers. These firewall rules are set based on the security policy of the network. In practice, proper implementation of firewall rules is a difficult task. Based on analysis of real configuration files, it has been shown by Wool [1] that in practical networks, a rule set of having more than 1000 items includes more than eight errors. To identify these errors, we need to use proper security validation and verification tools.

Another important area in network analysis is checking the service availability and survivability under different network conditions such as link failures and heavy loads. The term *availability* means readiness for correct services [2], *survivability* refers to a system's capability to fulfill its mission in the presence of failures or accidents [3]. There is a need for validating the reachability and the enforced security policies in an event of network failure. Such analysis helps to design a reliable network with failure recovery or to identify existing network configuration problems. Network failures can be an outcome of unstable wireless connection, temporarily overloaded link, or slow convergence of routing protocol. Failures on lower network layer may affect higher layer services. Therefore, implementation of proper network security has become an important, demanding and challenging task.

1.2 Problem Statement

The effect of link or device failures on a small network segment in a large dynamic network can propagate to other network segments and it is difficult to predict these

propagating effects. These propagating effects can change the communication paths, and hence the applied filters can be different. The newly applied filters can open restricted services or block the required services. Apart from packet filters, there can be filters to filter the routing updates. Therefore, there can be situations that some routers will not receive new routing updates; hence, some network parts will be unreachable. Since there are many combinations and dependencies, it is difficult to evaluate each scenario by generating test cases.

To overcome the above problems, the research is conducted to find an effective analysis method for evaluating reachability and security properties on computer networks configured with dynamic routing protocols.

1.3 Motivation

Modern computer networks have become more complex and they are required to provide various demanding services. Therefore, network administrators are compelled to use sophisticated network management tools to monitor the functionality and the performance of their networks. These networks are often required to promise high-level of availability; hence, the early detection or prevention of unwanted situations is desirable. On-line techniques, such as monitoring, logging and triggered notifications can help in case of a failure for administrators to become aware of the problem.

With the growing complexity of network designs, the tasks of configuration, implementation, maintenance and modification have become a challenge for network administrators. It can be widely seen that configuration errors of networks together with device failures are the reasons for most of the network outages.

The network designers often incorporate various techniques to minimize failures by configuring dynamic routing protocols. These dynamic routing protocols ensure to enable the services via the best available path. When the network is configured with dynamic routing protocols, the routers update their routing tables based on the current network state. Since the failures can change the communication paths and applied filtering rules, there can be open security threats, which are not visible in the running network state. Therefore, it is difficult to predict the reachability of different network states.

Research on analysis methods to evaluate the reachability and security properties under different network conditions has a potential industrial demand. Building a proper model that can predict dynamic behaviors of networks will be very useful to ensure the delivery of quality service under protection. Therefore, we are motivated to find a solution to this industrial need.

1.4 Significance and Objectives

The main objective of my PhD research is to build an effective model for dynamic networks and develop an analysis method to predict properties such as service reliability, security, and safety. Dynamic networks use intelligent routing protocols and in the case of a device or a link failure, consequent topology changes appear and response of the network can be different.

The research has two main components, namely, modeling and analysis. Following are the individual objectives of each part.

1.4.1 Modeling

Modeling is one of the key components on this research. Modeling phase can be further divided into two parts, modeling of network topology and modeling of device configurations such as routing and filtering. Modeling phase will be performed either to improve current available models or to invent new modeling methods.

The intended network model must be able to facilitate advanced configurations such as policy routing, tunneling and hidden paths. We could not find any existing model that could support these.

Another objective is to develop a model that is independent of the configured dynamic routing protocol or has less dependency on the modeling process. The model should be able to analyze the reachability and security properties from any source to any destination under any network state without rebuilding for each scenario.

It is also important to build a model to analyze the routing behaviors and the effects of routing filters. This type of model enables administrators to implement better routing policies on networks to avoid network isolations on device failures.

Then we need to embed the filtering rules into the model. This is archived by configuring the filtering rules on the devices. To model the filtering rules, we will consider format of Access Control List (ACL). The applied ACLs remain unchanged, if the network is configured with static routes. When a network is configured with dynamic routing protocols, and if a device fails, then different paths will be selected for the communication and the applied ACLs will be different.

Another objective is to research on a modeling method to accommodate traffic loads, transition delays, Network Address Translations (NAT) and Quality of Services (QoS). This is helpful to predict availability and quality of services after a network convergence.

Embedding these parameters to the model, we can make the model closer to the actual behavior of a dynamic network. All above considerations are to model packet changes, while propagating through devices; therefore, it is important to check the feasibility of modeling the applications and applications servers by using the same techniques.

The last objective in modeling phase is to automate the modeling process by reading on-line configurations.

1.4.2 Analysis

Analysis is the second major phase of the research, which contains several significant research objectives. This includes analysis of reachability, device configurations, routing, filtering and quality of services. We also look into an area where we can use verification and validation techniques to automate the analysis process.

One main significant area is to reduce the state space, the objective is to introduce and formally define an efficient new method for the representation of network states. Existing representation of the state space is by using the link states (up, down), this is exponentially growing with the number of links. Therefore, we need to research for a better representation of the state space.

Once the model is built, we must have an analysis approach to determine the reachability and security properties. The objective is to build a universal model that does not require rebuilding for each network state. When the network reachability is analyzed, then it may be checked for the QoS to ensure the required service level of delivery.

The analysis of network behaviors should be able to carry out under different configurations of routing protocols and combinations of them. This needs deep understanding of routing protocol behaviors and its redistribution process to populate the new updates in order to build the routing tables. This is a very significant area in routing analysis.

The analysis process checks a validity of a property in the network model. Therefore, it is advisable to use verification and validation techniques in the analysis process. The objective is to use satisfiability (SAT) base analysis techniques and build a framework for model checking.

Another consideration is to embed the model and the analysis process into a simulation tool. The objective is to build the simulation framework, which is capable of acquiring the network configurations, link states and QoS parameters from network devices and to build the model without any human interaction. This will be a very useful tool for the administrators and the main advantage is that they do not need to disturb the running network for the analysis.

1.5 Proposed Research Methodology and Plan

This section presents the outline of the research, research methodology and the research plan. Initially, we set the research boundaries to understand the major components of the research areas. The research is divided into four main sections, namely, filtering and routing, network modeling, configuration abstraction and analysis methods as shown in Figure 1.1. Then these main sections are further divided into subsections to make the research boundaries clear. This categorization gives us to work on the research independently in different sections in parallel, to keep the consistency among the research parts, and allows to concentrate and cover all parts of the scope of the research to achieve the main objective.

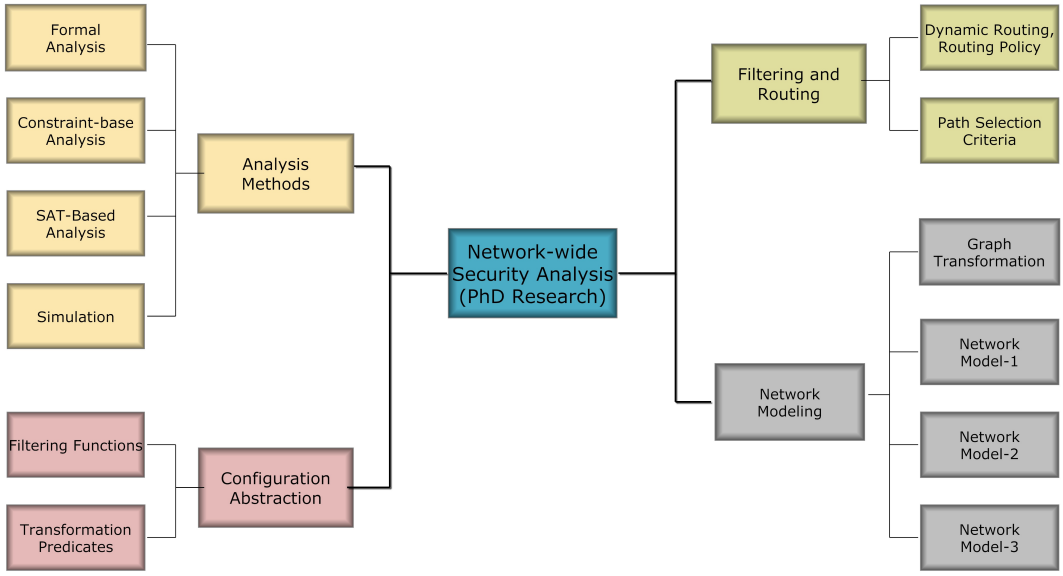


FIGURE 1.1: Scope of the research

The proposed research methodology and the plan for each section are discussed below:

1.5.1 Modeling

For the modeling of topology, a method based on graph theory, lattices and first order logics are incorporated. Subsequently, we are planning to use formal methods to define the semantics of networks and its dynamic behaviors. The intention is to develop a model independent of the configured routing protocol in the network or to limit the behavior of different protocols for a small number of variables, and hence planning to minimize the protocol dependency.

The main consideration in network modeling is to represent the network as a graph, which can represent all configuration scenarios. The plan is to use two different types of graphs, namely, hyper graphs and bipartite graphs to represent networks. Further, a separate model is to be developed by using forwarding devices and routing information to model the behavior of the dynamic routing protocols.

Our objective is to build three different network models (see Figure 1.1) for different analysis methods. These network models are using different types of network graphs; hence, to check the consistency between the network graphs, a graph transformation method should be engaged. The aim is to use graph grammar as the graph transformation method to transform the nodes, links and paths between the graphs. In addition, the graph transformation will be very useful to convert the identified potential communication paths during the analysis process from the graph models to the network diagram.

1.5.2 Filtering and Routing

To represent filtering rules, the format of a rule should be properly defined. Different vendors use different types of formats; hence, Cisco ACL format is considered for the illustration. Few challenging tasks when dealing with filtering rules are identification and elimination of the conflicting rules, effective representation of the IP addresses, and matching the IP header fields. To eliminate the conflicting rules, a proper algorithm should be developed. For the representation of IPs, available address schemes should be incorporated and for the packet matching process, Interval Decision Diagrams (IDD) will be considered. Most of the variable fields in ACLs are intervals e.g. network IPs, subnet masks, etc. Thus, matching process requires the interval matching and the use of IDD will improve the efficiency.

To incorporate routing into the model, the plan is to build a global table that will be valid for any network state. Therefore, rebuilding the routing tables for each instance will not be required. This global model will be used to check the reachability and evaluate the applied filters for any given network state.

ACLs are used to filter the data packets, there are special filters to filter the routing updates. These routing filters can be used to filter the routing updates from one network or not to populate the routing updates to other networks. This is a very significant area on modeling the behavior of dynamic routing protocols. The plan is to study the behavior of forwarding devices, method of generating routing updates, types of routing updates and routing update distribution methods of dynamic routing protocols. Once

the routing updates and the distribution process are modeled, we need to formally define the routing filters and incorporate them into the routing update process.

1.5.3 Configuration Abstraction

The aim of this section is to formulate and analyze the device configurations. This section has two parts. First, the device configurations need to be formulated. Under this, we focus more on building abstract functions to represent different device configurations such as QoS, Load, NAT etc. The considered research methodology is to use the restricted joint operator in relational algebra. The intention is to build a constraint data model, which is based on the filtering network model. Other than the device configurations, we are thinking of extending this approach to model some application servers.

In the second part, constraint based queries are developed. The correctness of device configurations is evaluated by using these queries under the configuration analysis process. Once the constraint data model is in place, the analysis process can be implemented to answer the questions such as paths satisfying a given property, which packets can be delivered on a given path, and evaluate these properties in abstract way. For an example, to check whether a property is satisfied under different conditions, transitional predicates need to be defined. Some device configurations change the packet headers, while traveling through the network (e.g. NAT), so the predicates should be built with the transformational properties to evaluate the input and the output packets. These predicates are then used for automating the analysis process especially during the model checking and implementing the simulation tool.

1.5.4 Analysis

The aim of the analysis section is mainly to evaluate the reachability and the security properties. Different analysis methods are incorporated for analyzing different network properties such as reachability, routing, device configurations and filtering rules. We are considering to involve different methods for each analysis process. The plan is to perform the analysis using formal analysis, constraint-based methods, simulation tools and SAT-based techniques. Each has its own advantages, disadvantages and limitations, which will be detailed under relevant analysis chapters of the thesis.

For the reachability analysis, the plan is to use unified model defined by Xie et al. [4]. Unlike Matousek et al. [5], our analysis does not pre-compute all possible routing configurations in order to verify the network reachability. In the analysis approach, we

categorize links as Critical Points (CP), Universal Points (UP) and dependent points. Class of CP contains the links that are essential for the given communication, while the class of UP contains the links that are not required for the given communication. This analysis approach will eliminate the trivial cases and only the failures of remaining links will be analyzed in detail. Therefore, the overall computational time can be significantly reduced.

To reduce the computational cost, reduce number of iterations, eliminate rebuilding the model for each network state, and represent the network states in compact way, the link states that have the common network behaviors are grouped together. From this approach, the state space can be significantly reduced and the state space explosion problem can make less evident during the analysis process. Grouped network states combined with path selection criteria is used to analyze the reachability.

For the constraint-based analysis, the packet transformation based on the device configurations should be evaluated against the constraint queries. Since this contains logical operations and predicates, the aim is to develop a program by using Prolog for this analysis process.

The considered analysis approaches involve static models; hence, we are planning to use static analysis to automate the analysis process. During the analysis, reachability and security properties for a given network state are checked in the static model. Since it is a model-based verification process, the intention is to use an existing model-checking tool to automate the analysis process. Our aim is to develop the frame works for SAT-based method to evaluate firewall rules in dynamic networks. The intension is to encode the problem as SMT problem and to generate counter examples in the case of security violation.

The last part of the research is to develop a simulation tool. The approach is to embed the analysis process into an existing simulation environment. The identified simulation environment is $OMNeT^{++}$. The ideal implementation should read the configurations on-line from the devices, build model and perform the analysis in real time.

1.6 Contribution

The research on Network-wide Security Analysis has significantly contributed in each section categorized under the scope of the research shown in Figure 1.1. In the modeling phase, the thesis introduces two new approaches to construct different types of network graphs, and an effective way to model the device configurations and model the routing update process. In addition, the thesis shows how these models can be used for different

types of analysis. In the analysis phase, we have a major contribution on reachability analysis and analyzing the security properties. The contribution of each phase is detailed in following paragraphs.

The filtering network model is a new network introduced from this thesis. This approach also contributes by introducing a novel method to check the end-to-end connectivity in a network[1]. From the given device configurations and the network topology, the constraint-based analysis is able to determine the availability of services in a network. We also capture the security aspects by considering firewall rules along network paths from the traffic originator to the intended destination. Using this analysis, it is possible to find *backdoor* or *hidden paths* [6] in a network that can be used for the unauthorized access to network services.

The described methodology and the plan are to determine reachability by using a centralized table. This method works efficiently on analyzing the network reachability and security properties in dynamic networks. It does not require pre-computed routing tables for each network state. It is also shown that the network reachability and security properties can be evaluated under any network state without rebuilding the centralized table. This is a unique innovation contributed to the area of path selection in networking. The analysis method was tested by using java program.

The method used to reduce the state space is based on grouping states that have similar network behaviors. These groups of network states are referred as the General States. This is also a unique innovation contributed to the area of network modeling.

When analyzing the reachability under different device failures, we need to eliminate the obvious cases; analysis of devices that are always required for the communication and the devices that have no effect for the communication. The devices are categorized into two sets, namely, critical points and universal points. If the devices or the links are a subset of universal points, it is possible to ignore the effect of the potential failure. If a device or a link is a member of the critical points, without further analysis we can predict that there will be no available path for the communication. Categorization of devices into sets contributed to improve the analysis process.

The introduction of constraint-based approach for modeling and analyzing device configurations allows handling complex queries. These queries can be used to check the packet qualities, nature of packet transformations, and the properties of tunneling and hidden paths. This is also a key contribution of this thesis. Further, this analysis is automated by using Prolog language and the developed tool is capable of handling complex queries efficiently.

[1] In literature, this problem is sometimes called border-to-border availability.

Modeling and analyzing routing information base by using forwarding device model contribute an alternative way to analyze the routing without simulating the routing protocols. This also introduces a framework to build the routing tables from routing updates. As the method is based on a unified network model that captures both routing and filtering, it can be further used to verify i) the service availability and ii) proper implementation of security policies on the network. This also gives a better picture for the network designers to properly configure routing and to avoid unwanted flooding of routing updates on networks.

In firewall policy analysis, a complete framework, which is capable of evaluating the network security policy against the configured filtering rules was developed. This was achieved by converting the problem into a SMT problem. The analysis is capable of generating the most-general counter example in the case of a security violation. The development of analysis method by using the SMT tool is the key contribution in the area of firewall policy analysis.

The analysis with simulation tool includes i) extension of formal analysis for network reliability. In addition to our first paper [7], this ii) introduces notion of *criticalness* as an important property related to survivability and shows how to use it to detect possible weaknesses in a topology. Another contribution is iii) integration of formal analysis techniques into a simulation tool. A program has been developed to analyze reachability, convergence delays and packet loses. This was developed by using the $OMNeT^{++}$[2] simulation environment.

In summary, the main contributions of this thesis are introducing a novel approach in formal modeling, analyzing the reachability and security properties of networks configured with dynamic routing protocols, and an effective method to reduce the state space. In the area of modeling, implementation of filtering network model, modified topology table and general state are the key innovations. In analysis, the constraint-based analysis shown to analyze the device configuration and answer complex queries are the key innovations. In the area of routing, an extension of an existing method is presented to model dynamic routing protocols and filtering of routing updates. In firewall policy analysis, the implementation of SAT based decision procedure and the framework to generate counter examples are the key achievements. Finally, we were able to develop a simulation tool in $OMNeT^{++}$ environment to analyze reachability and associated delays.

[2] see http://www.omnetpp.org

1.7 Structure of the Thesis

This thesis consists of nine chapters. The content of the thesis is structured as introduction, literature survey, modeling, analysis and the conclusion. The overall thesis structure is shown in Figure 1.2.

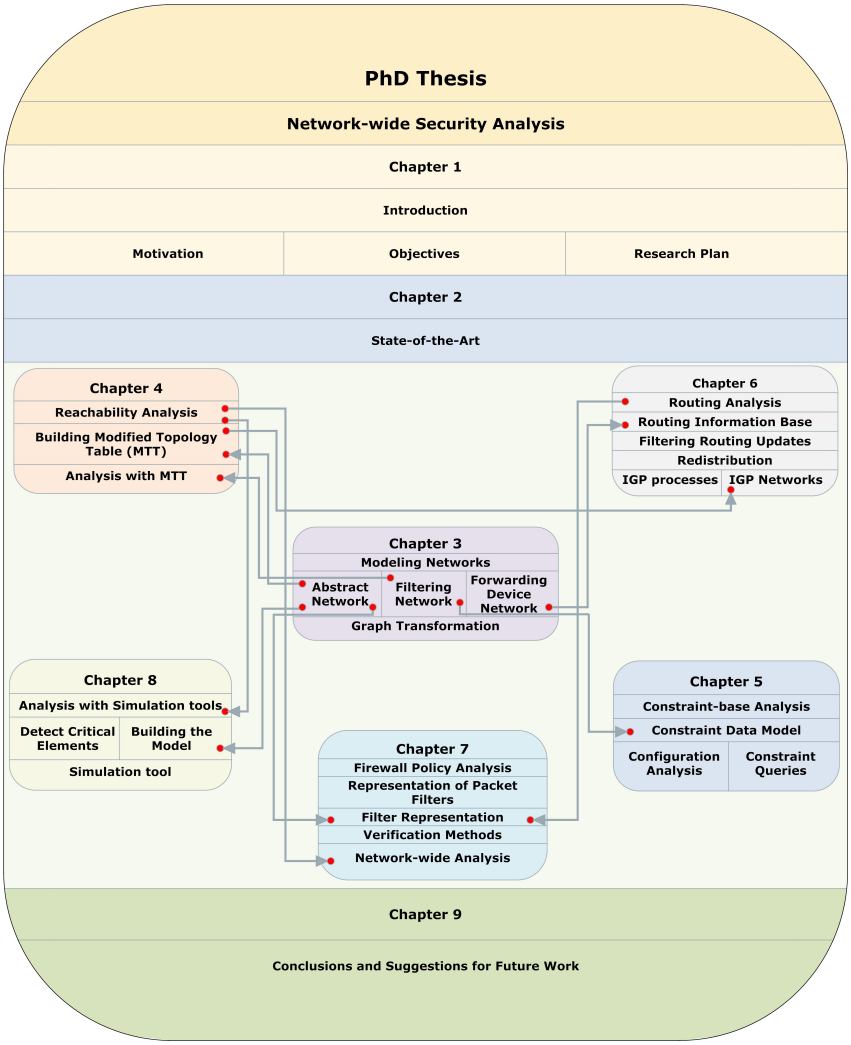


FIGURE 1.2: Structure of the Thesis

The first chapter, Chapter 1, gives an overview of the research including the problem statement, objectives and the contribution. The second chapter, Chapter 2, reviews related work regarding dynamic network analysis. This chapter also gives an overview and the terminologies used in this area.

Third chapter, Chapter 3, presents different network modeling approaches and develops models that are capable of representing network topology, routing, and filtering rules.

Three different types of network models are built for analyzing reachability, device configurations and routing. Further, Chapter 3 represents the most important and the core work of this research, which links to all other chapters. The linking of the related models, which built under this chapter, to the relevant analysis chapters is depicted in the central part of the Figure 1.2.

Next five chapters explain different analysis methods. Chapter 4 presents our core analysis method and describes the reachability analysis using a centralized table. Further, it shows a method to reduce the state space, where the analysis can be performed with less iteration. Chapter 5 formulates configurations such as ACL, QoS and NAT by using abstract functions and describes a constraint-based method to evaluate the device configurations and complex queries on networks. Chapter 6 discusses on routing analysis and Chapter 7 details on evaluating firewall policies by using SAT-based analysis techniques. The implementation of reachability analysis by using a simulation tool is shown in Chapter 8.

The last chapter of the thesis, Chapter 9, discusses obtained results and concludes this research work. Further, it shows the avenues for the potential future work, which can extend and improve the presented research work from this thesis.

Chapter 2

State-of-the Art

The main objective of this chapter is to give an overview of the terminologies and to describe the related research work performed by other research groups in the area of network modeling, analysis of routing and filtering rules, and applicability of verification methods. The related work is described in the form of their authors' approach, current status, results, and limitations. As stated in Section 1.5, some of our research areas are extensions or improvements of others research. In this case, clear boundaries have been set to separate their and our new contribution. When a new approach is used, clear comparison has been made to identify the advantages of our approach.

Configuration of active network devices is a distributed program that controls a behavior of each network node. Correct configurations should satisfy the intended network functionality. Validating correctness of network design is difficult and cannot be achieved as a single task. There are many requirements that a network should meet.

Before we dwell on a survey of the current methods for validating network configuration, we give an overview of requirement classifications. Usual requirement taxonomy known from software engineering splits requirements into two classes as functional and non-functional requirements.

Functional requirements are determined by the basic assumptions derived from design goals of a network. The fundamental functional requirement of a network is to deliver data. Another example of functional requirement is the ability of a network to provide diagnostic data, e.g. in the form of logs from network devices or Netflow records.

Non-functional requirements are known as dependability, which consists of the following categories:

- Availability – the degree to which network service is in a specified operable state.

- Reliability – the ability of a network to perform its required functions, e.g. data delivery.
- Safety – the protection against damages, losses, harm, or any other undesirable consequences of network failures.
- Confidentiality – the absence of unauthorized disclosure of information from network communication.
- Integrity – the protection against improper network configuration modifications.
- Maintainability – the degree that allows administrator to perform network modification and repairs.
- Quality – the capability of a network to provide a specified level of availability and reliability for selected functions or services.

Network functions can be implemented on different layers. For instance, a reliable data delivery is possible, if network layer can find a viable path from source to destination by running distributed computation of routing database and transport layer can implement TCP session for compensating packet losses. The validations of the requirements are difficult, since the underlying concepts of each model are different. While there is no method that would perform complete analysis, there are several methods that are able to partially validate network requirements.

Network reachability can express certain dependability requirements. For instance, service availability depends on the possibility to reach the service by clients in all operable states. A protection against an unauthorized access can be also validated by checking reachability property. Only the authorized clients should have connections to the relevant services.

In IP networks, network reachability is mainly considered as a function of the network layer. In this layer, packet forwarding, packet filtering, and packet transformation express the network functionality that provides the data delivery function. This is only a partial view, since other layers also significantly contribute to the reachability.

2.1 Network Reachability

This section lists the research work done in modeling of networks and different approaches used to analyze reachability, routing behaviors and filtering rules.

Network reliability provides an assumption on data delivery in a network. The expected reliability requirements are dependent on the network conditions. In this section, we overview methods that aim at checking reliability of data delivery considering different network conditions.

A network condition is a collective state of network devices and communication facilities. For instance, depending on the level of abstraction, the state of a communication facility can be represented by the link capacity. Nevertheless, even simpler model based on failures is useful for the analysis of network reliability. In our model, we consider that network condition is expressed by operational states of devices and communication links. Iannaccone et al. [8] performed an analysis for the effects of link failures in an IP backbone networks. The impact of a link or a device failure in an environment with redundant paths is visible during the convergence period, which is the time needed by dynamic routing protocols to compute routing information. According to [8], it is evident that failures are very common in typical networks. Moreover, majority of failure events of short durations (about 50% of total failures) are caused by overloaded routers. During this short period, the routers are unable to function as expected, e.g., failure to maintain adjacency between routing processes, which can be identified as a device failure by other nodes.

An early report on formal analysis of end-to-end network reachability was presented in 1997 by Guttman. He defined a formal method to compute a set of filters for individual devices and constructed a global security policy in his research work [9]. To achieve feasibility, the network was abstracted and represented only with the network areas and the border routers in his model. This natural decision mirrors the real situation, as internal routers do not participate in data filtering. Similarly, data flow model was defined in terms of abstract packets, which were described by abstract source addresses, abstract destination addresses and service types. Then an algorithm was developed to compute a feasibility set of packets that can pass all filtering rules along a path. The employed abstract packet description makes the procedure practically feasible and efficient. The presented method is able to compute network reachability on a network model that counts only the filtering rules. A prototype of the algorithm was done in a Lisp-like language. The program is able i) to generate distributed firewall configuration based on global security policy, and ii) to check the correctness of a policy implementation against its specification. Global security policy has the form of reachability requirements that define a set of services, servers providing these services and clients authorized to access those services.

Later, Guttman and Herzog extended the previous approach by incorporating IPSec gateways [10] and then both approaches were combined to a uniform framework [11].

This method deals with a simple network model, which is represented as a bipartite graph where nodes are routers and networks, and edges are interfaces, which have associated filters. For modeling of filters, an abstract packet representation is defined. A packet is described by selected fields from the IP-header. A notion of *trajectory* was introduced for packet-path pairs to define possible packet reachability within a network. Then security policies will remain unchanged, if there is no occurrence of unwanted trajectories in the network. A security policy is a property of trajectories and can be defined as a set of permissible packet-path pairs. For an example, a typical policy is read as, if p was ever in area A and later reaches area B, then p should be a TCP packet destined for port 25 and with destination address of a valid SMTP server. It is claimed that realistic policy can be expressed in this way using only two-area statements. The authors implemented an experimental program called Network Policy Tool. For efficient evaluation, network and host IP addresses are not treated explicitly, but a symbolic representation is used. A symbolic name is assigned to a range of IP addresses that are treated in the same way by the filtering rules. Filtering rules of an access control list are converted to a set describing the meaning of the whole access list.

The analysis process developed by Guttman et al. [10], checks IPSec configurations to ensure the packets that should be protected are not misrouted to public networks without applying protection mechanisms. If packets are protected, then their trajectory is evaluated to ensure that packets will reach the intended IPSec gateways. An enriched system model has been introduced to define the locations of IPSec processing. These correspond to the device interfaces that represent inbound and outbound processing points for packets. The evaluation methods employ a model that describes packet delivery by a state machine. The states determine packet locations in the network. The transition relation reflects the packet operations, namely, create, discard, move, prefix, and pop. First three operations are related to packet delivery in the network. Prefix and pop operations manipulate with packet headers by adding a new header or removing the outermost header respectively. The notion of *trust set* was defined in order to express trusted network locations where packets may be delivered. Trust sets are dependent on security goals and contain collection of sub-networks that are connected via IPSec tunnels. Trust sets have defined boundaries that are represented by devices, which are capable of IPSec and packet-filtering operations. An evaluation method is implemented in CIAC tool to check trust sets and boundaries against security goals based on the provided configuration.

The static analysis of IP networks was initially addressed in research work by Xie et al. [4]. They defined a framework that is able to determine lower and upper approximations on network reachability. An underlying network model uniformly describes filtering rules, dynamic routing and packet transformations, e.g. NAT or ToS remapping. The

method computes symbolically a set of packets that can be carried by each link. By combination of these sets along all possible paths between two end points, it is possible to determine the end-to-end reachability. The upper approximation determines the set of packets that could potentially be delivered by the network, while the lower approximation determines the set of packets that could be delivered under all possible forwarding states. In their paper, the authors also presented a refinement of both upper and lower approximations by considering the effect of dynamic routing. It was shown the steps to convert dynamic routing information to filtering rules in a unifying network model. Algorithms were presented for this conversion and for estimating lower and upper bounds on reachability. Finally, the problem of reachability analysis was discussed in larger context by exploiting possible applications, e.g., analysis of failure scenarios, validation of design patterns and combination with dynamic information. We have used a modified version of their approach to build the model of forwarding devices and routing information in Section 3.4.

Narain, Talpade and Levin have shown in several papers (c.f. [12], [13], [14]), how to develop an approach to validate network configurations against end-to-end requirements. They described a design of configuration validation system called IP Assure (formerly *ConfigAssure*). Validation of requirements is based on application of logical programming, constraint solving over finite domains, and SAT. The benefit of configuration validation was illustrated on a network with decentralized administration, which is inherently vulnerable to configuration errors. The analyzed properties are GRE and IPSec tunnels, consistent addressing scheme, OSPF and BGP basic configuration, and MTU settings. ConfigAssure tool consists of an acquisition subsystem for extracting configurations from components, a requirement library that collects typical end-to-end requirements, a specification language for writing requirements specifications, and an evaluation component that checks the configuration against the active requirements. Extracting information from configuration file is performed by ANTLR parser. It skips parts of the configuration that have no meaning for the analysis by applying a special rule. The data are then inserted into database tables.

IP Assure can handle several categories of requirements. Requirements on integrity of logical structures that are checked by IP Assure are, IPSec and GRE configuration, HSRP configuration, iBGP peering, OSPF areas and MPLS tunnels. The connectivity requirements evaluated by IP Assure are IP, VLAN, GRE, IPSec, BGP and MPLS. IP Assure can check reliability requirements by verifying an absence of single point of failures in IP network, existence of multiple OSPF area-border-routers and replication of IPSec tunnels. IP Assure verifies the implementation of IPSec configuration and ACLs as part of security requirements. The tool also evaluates performance requirements by

checking whether the policies on all routers are identical. Users can specify the required security properties to evaluate the configurations.

The implemented evaluation system by Narain et al. [14] employs usual graph algorithms. They used symbolic method for checking firewall configurations. The extended configuration validation employs constraint representation over finite domains and constraint solvers, e.g. Kodkod. A Prolog program was used to check the requirements. It is sufficient to check whether the requirement is satisfied by the configuration. Moreover, ConfigAssure will offer suggestions for configuration corrections, if the configuration checking fails. This task was implemented with the help of a constraint solver.

Al-Shaer, Marrero, El-Atawy and El-Badawy (c.f. [15] and [16]) described a tool called *ConfigChecker* that allows general reachability and security property-based verification. The ConfigChecker maintains a description of a network dynamics by modeling the forwarding process as a state machine. Each state is defined by occurrences of packets at network locations. This approach uses a symbolic representation. Reachability requirements are expressed in CTL and evaluated by a symbolic model checker. The model of a network is presented as a finite state machine and its actual state is determined by locations of packets in the network. Thus, a state is encoded as $\sigma : packet \times location \rightarrow \{true, false\}$, where *packet* is an abstract specification of a packet. The behaviors of devices are defined in terms of packet transformation, e.g., NAT changes information in the packet header and its location in the network. This change is described as a Boolean formula. Using this approach, it is possible to uniformly represent the operations of network devices as a (large) Boolean formula. By exposing more information into the packets, one may encode encapsulation transformation as well, e.g. IPSec takes an incoming packet and put it inside a new IPSec packet, which requires to have a packet model consisting of outer and inner header fields.

This method was implemented by using BDD for encoding Boolean formula that represents transformation relations. To cover all information fields in a basic IP packet header, 104 Boolean variables need to be employed. Additional seven variables are required for expressing the IPSec information. A standard CTL model checking algorithm can evaluate the required end-to-end reachability properties and direct CTL queries verify the basic reachability properties, for an example, if packet p reaches location l, then it can be expressed as $EF(location = l)$. The presented security analysis aims at verification of the authorized access and takes all allowed flows between two locations as the input. Then the configuration expressed in terms of the state machine is evaluated against the set of requirements.

Due to the expressiveness of CTL, it is possible to verify other non-trivial properties, such as:

- absence of routing loops,
- shadow or bogus routing entries, which stand for routing decisions that will never be applied,
- integrity of IPSec tunnels, including nested or cascaded tunnels, and
- absence of backdoors or broken flows that may happen after routing changes.

The implemented prototype was tested on more than 90 networks, and its performance was evaluated. Presented results demonstrate the feasibility of the method in terms of time and space requirements. Building a model of network with thousands of nodes requires tens of seconds and grows linearly.

An approach based on Xie et al. [4] for analysis of end-to-end network reachability was suggested by Bandhakavi et al. [17]. They separated routing and filtering to simplify the network model. Checking end-to-end reachability is a process that consists of four phases:

- In Model Instantiation phase, the network model consisting of device models is populated with configuration information.
- In Route Calculation phase, the route advertisement graphs are constructed. The computation is performed according rules specified by each participating device. Then the routing information base is filled with computed routes.
- In Route Analysis phase, the information on possible routes is used to calculate all end-to-end connectivity.
- In End-to-End Validation phase, the end-to-end reachable paths are checked against the requirements. Any violation of a requirement is reported together with suggested fixes.

The effect of routing to end-to-end reachability is included in the analysis by computing route graphs. These graphs are computed for every target network. To compute route graphs the approach described in [4] is employed, which means that rules of routing protocols are used in routing propagation algorithm to model the routing information flow. After reaching a fix-point, the routing information base contains all routing data. From route graphs, all paths for the given end points are selected and, on these paths, the applied filtering rules are evaluated with respect to the given requirements on packet delivery. The method is extended for round trip flows when state-full firewall rules need to be considered.

Bera, Dasgupta and Ghosh (see [6], and [18]) defined a verification framework for filtering rules that allows one to check the correctness of distributed ACL implementations against the given global security policy and to check reliability (or fault tolerance) of services in a network. To check the correctness, initially, the filtering rules are translated to assertions represented as Boolean formulas. Then the Boolean formulas together with the translation of global security policy are sent to SAT solver. In case of an inconsistency, the SAT solver may produce a counter example that helps administrator to debug ACL rules. To check the reliability, a framework was built to check whether the ACL rules are consistent with the given global security policy. A policy is understood as a description of service availability with respect to defined network zones. First, the method computes the network access model, which is a directed graph with ACLs assigned to its edges. Then, the Service Flow Graphs (SFGs) are generated for the services to be checked; such as, SFG for ssh traffic. An SFG is a sub graph of network access graph. Based on the network access graph the fault analysis is performed by computing values of minimum cut in all SFGs. These values represent the number of link failures that can be tolerated.

Gan and Helvik [19] employed probabilistic methods to reduce the space of possible network states. They used stochastic activity networks to describe failures of network components and other dynamic issues. This restricts the state space to the subset of the most probable situations. Another algorithmic framework based on probabilistic calculations was discussed by Michael et al. [20]. They presented a framework for the analysis of ingress/egress unavailability and link congestion. The framework is able to deal with three causes, namely, failures, changes of user behaviors and rerouting.

2.2 Verification of Security Policies

This section mainly lists the related work performed to automate the evaluation process of filtering rules. This research work gave us the initial motivation to research on verification techniques in network security. Further, these research works have proved that verification techniques can be applied to check the correctness of the implemented network security.

Packet-filtering firewalls are supposed to enforce network security policies by inspecting the packets flowing through them. These filters are controlled by configurations. These configurations contain low-level rule-bases and they are implemented according to high-level security policies. A significant effort is made to develop a method verifying whether the firewalls correctly enforce the defined security policies. As firewalls can contain thousands of rules, the methods presented in this section aim at efficient firewall analysis.

A firewall configuration is usually defined by a collection of deny/permit rules. A rule, which applies when processing any given packet is selected by first match semantics. Thus, a fundamental problem is to verify that the intended rule will be selected for a specified class of traffic. Firewall configuration is often adjusted to new demands by adding new rules, removing old rules, or modifying existing rules. It is not unusual that such modifications introduce internal conflicts within the relevant firewall rule-base. These conflicts are classified according to their severity, which is defined by a relation between conflicting rules. A rule may hide another rule or overlap with other rule performing the same or a different action. The algorithms that are devised to find conflicting rules and to correct firewall rule-bases are surveyed in below part.

Bartal et al. [21] introduced a tool for generating firewall rule-bases from a security policy design. A security policy design is a high-level abstract description of security requirements. The design is an instance of an entity-relationship model. An entity-relationship modeling framework is built around a central concept of *Role*. A role defines a communication capability, e.g., a web server may accept any HTTP request from any computer on Internet. Individual roles may be aggregated in role groups that enable to specify service availability for groups of hosts. The security policies are written in a simple security policy design modeling language. The model compiler takes the specification of a security policy made by a security administrator and produces an appropriate firewall configuration files. A rule-base generation is done in two phases. First, a centralized rule-base is generated. Second, the centralized rule-base is adapted to each firewall device in the network by analyzing the network topology specification. This method is efficient since the security policies can be expressed as triples, each of them consisting of a group of source locations, a group of destination locations and permitted services. With direct analysis of permitted services, it is possible to infer what ports should be permitted. By analyzing the topology and mapping locations specified in roles to network locations, it is possible to deduce IP addresses and the assignment of filtering rules to particular network firewalls.

Al-Shaer and Hamed analyzed anomalies of firewall rule-base in [15]. They also presented algorithms that unveil certain inconsistencies and conflicts by performing intra-firewall and inter-firewall analysis. The main declared contribution is the analysis of distributed firewalls. To classify anomalies, several kinds of relations among rules are defined, namely, completely disjoint, exactly matching, inclusively matching, partial disjoint, and correlated. A firewall policy is represented as a policy tree that enables to discover anomalies by checking the coincidence of paths of any pair of rules. Rules where the paths do not coincide are disjoint. In case of a coincidence, one needs to apply an

anomaly discovery algorithm, which determines the relation between rules by comparing individual fields. Inter-firewall anomaly discovery is a complicated task. Anomalies between different firewalls may contain:

- shadowing anomaly, if an upstream firewall blocks the traffic accepted by a downstream firewall,
- spuriousness anomaly, if an upstream firewall permits the traffic blocked by a downstream firewall,
- redundancy anomaly, if a downstream firewall denies the traffic already blocked by an upstream firewall,
- correlation anomaly, which can lead to shadowing or spurious anomaly.

An inter-firewall anomaly discovery algorithm reveals anomalies in a collection of firewalls. The algorithm considers all paths between analyzed domains as its input. For any path, it is necessary to analyze all firewalls along the path. First, firewalls are analyzed for intra-firewall anomalies. Then, a policy rule tree of the first firewall on the path is created. This tree is enriched by rules of all consecutive firewalls. Then the rules that apply to processed path are marked. Finally, the unmarked rules are added at the end and reported as irrelevant. Any anomaly detected while adding new rules into the base policy rule tree is reported as a warning or an error. The evaluation of presented algorithms demonstrates their efficiency on real scenarios.

The detection of vulnerabilities of hosts and their protection against network attacks were elaborated by Tidwell et al. [22]. Another two related works were published by Zakeri et al. [23], and Shahriari and R.Jalili [24]. Ou et al. [25] defined reasoning rules that express semantics of different kinds of exploits. The rules are automatically extracted from the OVAL scanner and the CVE database [26]. This approach has been implemented by using Prolog and the method is named as automatic deduction of network security.

Yuan et al. [27] developed a tool called FIREMAN, which allows to detect configuration errors in firewall settings. The FIREMAN performs symbolic model checking of firewall configurations for all possible IP packets and along all possible data paths. The underlying implementation depends on a BDD library, which efficiently models firewall rules. This tool can reveal intra-firewall inconsistencies as well as configuration errors that lead to errors at inter-firewall level. The tool can analyze a series of ACLs on paths for end-to-end connectivity, thus offering network-wide firewall security verification.

Pozo et al. [28] provided a consistency-checking algorithm that can reveal four consistency problems, namely, shadowing, generalization, correlation and independence.

Liu [29] developed a method for formal verification and testing of (distributed) firewall rules. Gouda et al. [30] developed a method for verification of user provided properties. They represented firewall rules in a structure called firewall decision diagram (FDD), which takes firewall rules as an input and convert it into a verification algorithm. Another input is a property rule, which describes the property that needs to be checked, e.g. description of a set of packets that should pass the firewall. By a single traversing on an FDD from the root to a leave, it is possible to check the validity of a given property.

Jeffrey and Samak [31] aimed at analysis of firewall configurations using bounded model-checking approach. They focused at reachability and cyclicity properties. To check reachability, it means to find for each rule r a packet p that causes r to fire. To detect cyclicity of a firewall configuration, it means to find a packet p that is not matched by any rule of the firewall rule set. They implemented analysis algorithm by translating the problem to a SAT instance and showed that this approach is efficient and comparable with tools based on a BDD representation.

In summary, this section briefly presented the research background and the applied research approaches by other research groups. The following chapters will detail on our research work and the corresponding references to other approaches have been made as and when they are used.

Chapter 3

Modeling Networks

This chapter shows building three different types of network models by using three approaches. To analyze the network behaviors, the network has to be converted to a form where we can extract network properties. This conversion is the first part of the modeling phase. There are different ways we can model networks. As stated in the scope of the research, our objective is to use different analysis methods to evaluate the reachability and security properties of dynamic networks. To apply these methods, suitable network models should be defined.

In the first approach, the network topology is converted to a digraph and creates an abstract network graph. The graph has two types of vertices. First type of vertices represents the devices. We call these devices as forwarding devices, since they forward the packets. These vertices have associated configuration functions, which extract the information from the device configurations that have effects on network reachability, e.g. filtering, and packet translation. Second type of vertices represents the communication facilities. Communication facilities are the links. Then the abstract network graph is combined with other network parameters to build the *abstract network model* (ANM). This ANM is used for reachability analysis as shown in Chapter 4.

In the second approach, we propose a model that describes the effects of network configurations in terms of packet flow filters and transformations. The cumulative effect of packet processing in a forwarding device can be represented by transformations associated with every pair of inbound and outbound interfaces. This model is suitable for reachability and security analysis considering standard network layer security policy implementations, e.g. access control lists or network address translation. This network graph is called filtering network graph, which also contains two types of vertices and has the properties of bipartite graphs. By adding more network parameters, we can build

the *filtering network model* (FNM). This FNM is used for constraint-based analysis as shown in Chapter 5.

In the third approach, the abstract network graph is combined with routing information base. This method is capable of modeling advanced routing configurations, and combinations of different routing instances and protocols. The detailed routing analysis by using this model will be shown in Chapter 6.

The simplified sequence of events during packet forwarding is as follows: i) forwarding device accepts a packet at one of its interfaces, ii) performs preprocessing of the packet according the rules associated with this interface, iii) switches the packet to an outbound interface based on the information in its routing table, and iv) performs post processing according to the rules associated with the outbound interface.

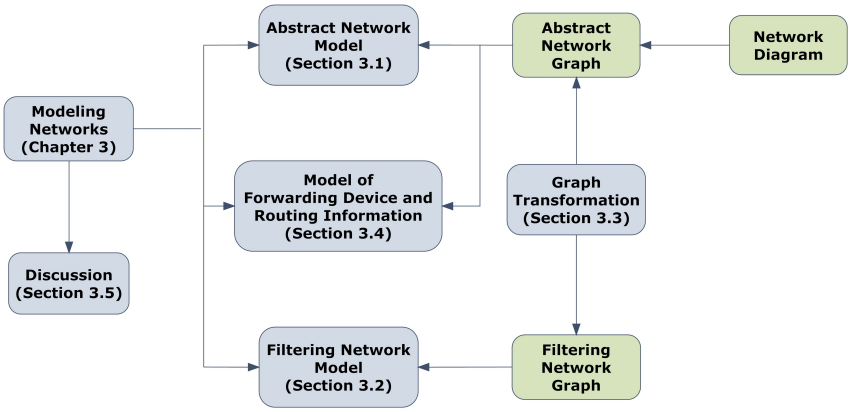


FIGURE 3.1: Structure of Chapter 3

This chapter is structured as shown in Figure 3.1. Section 3.1 defines an abstract network model by specifying abstract network graph (N_A) and informally describes the process of building N_A by using network topology information. Section 3.2 describes a filtering network model by using filtering network graph (N_F) and shows how to build this graph from N_A. Section 3.3 defines graph transformation between N_A and N_F graphs, and we use graph grammar for this transformation. Section 3.4 explains building a different type of network model using routing information bases. Last part of this chapter, Section 3.5, discusses differences of these models and possible applications in analysis of reachability and security properties.

Introduction to graphs

Let's represent the network topology by using graphs. A graph is formally defined as follows:

Definition 3.1 (Graph G)**.** Let $E(V) = \{\{u, v\} \mid u, v \in V, u \neq v\}$, a pair $G = (V, E)$ with $E \subseteq E(V)$ is called a **graph** where V are the finite number of **vertices** of G, and E are the finite number of **edges** of G.

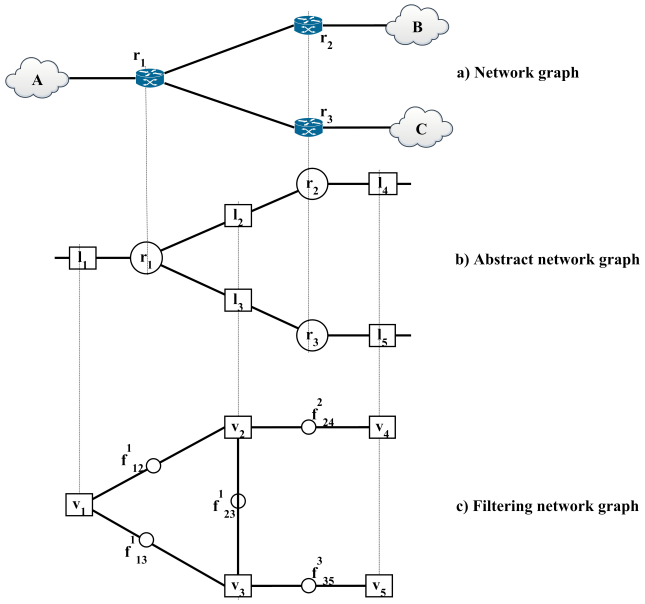


FIGURE 3.2: Types of network graphs

Abstract network graph $N_A = (\{r_1, r_2, r_3\}, \{l_1, l_2, l_3, l_4, l_5\}, E_A, \delta)$ is the source of transformation to filtering network graph $N_F = (\{v_1, v_2, v_3, v_4, v_5\}, \{f^1_{12}, f^1_{13}, f^1_{23}, f^2_{24}, f^3_{35}\}, E_F, \gamma)$. Router r_1 is transformed to three filters denoted by $f^1_{12}, f^1_{13}, f^1_{23}$. Similarly, router r_2 is transformed to f^2_{24}, and router r_3 is transformed to f^3_{35}. Hidden paths are not shown in this picture

A network diagram (network graph N) is a graphical representation of nodes, links and their interconnections. This is used to build two types of graphs. First graph, abstract network graph N_A is built by using network N and represent it as a digraph. Second graph, filtering network graph N_F is built by considering the network devices as collection of filtering nodes. This has the form of bipartite graph.

For modeling routers, switches and firewalls are considered as forwarding devices and Ethernets, leased lines, wireless links and microwave links are considered as communication facilities. Examples of a network diagram N, its abstract network graph N_A and the filtering network graph N_F are shown in Figure 3.2.

3.1 Abstract Network Model

This section details the process of building the abstract model of a network. This work was published in our research paper [32]. First part of this section formally defines the

network and builds the abstract network model by using graph N_A. It will be then used for the reachability analysis process as described in Chapter 4.

For abstract modeling, graph N_A is used and an example of N_A is shown in Figure 3.2.b. The vertices of type L are represented as squares and the vertices of type R are represented as circles.

To build the abstract network model, the combination of techniques introduced in Xie et al. [4], and Christiansen and Fleury [33] are used, but a different type of graph is incorporated. The network is regarded as a directed hypergraph where vertices are routing devices and edges are the communication links. These communication links are referred to as communication facilities. In real networks, there are other network devices then routers. However, every end-point device, such as PC or Web server, can be represented using a router with one interface. Formally, our *abstract network graph*, will be built as below:

Definition 3.2 (Abstract Network Graph)**.** $N_A = \langle R, L, E_A, \delta \rangle$, where

- R is a set of R-vertices that represents intermediate devices,
- L is a set of L-vertices that represents communication facilities, which are the links connecting intermediate devices,
- $E_A : R \times L \cup L \times R$ is a set of edges, and
- $\delta : R \to \mathcal{C}$ is a function assigning to each node that maps device into configurations $\mathcal{C}$.

The model of two channels over one physical link enables modeling asymmetric communications. Every physical link between two adjacent devices R_i and R_j is a pair of channels $l_{ij} = \langle R_i, R_j \rangle$ and $l_{ji} = \langle R_j, R_i \rangle$.

The above description of the network is static. It only describes network topology and the assignments of configurations to network devices. This description is then combined with the following network definitions to build the abstract network model.

Next-Hop function enumerates all adjacent forwarding devices for a specified device. Specifying a link as another argument, the function returns the connected device with respect to the specified link.

Definition 3.3 (*Next-Hop*)**.** Function $NH : R \times L \to R$ returns the connected adjacent device for a given device and a link. Formally, $NH(R_i, l_{ij}) = R_j$ where $R_i \overset{l_{ij}}{\to} R_j$, $R_i, R_j \in R$ and $l_{ij} \in L$.

Each link is associated with its cost, which is a metric used by routing protocols. The shortest path is the path that has the smallest accumulated cost among all possible paths.

Definition 3.4 (Cost Function)**.** For a given link $l \in L$, $C(l) : L \rightarrow \mathcal{N}$. In network terminology, it is called a metric.

Following record type is used to describe a packet. For simplicity, it consists only of five fields. If necessary, this can be extended with other fields.

Definition 3.5 (Packet)**.** $\langle\, protocol : \{ip, tcp, udp\}, src\_adr : IP,\ src\_port : (0 \ldots 65535),\ dst\_adr : IP,\ dst\_port : (0 \ldots 65535)\ \rangle$, where IP $= \{a_1.a_2.a_3.a_4 : a_i \in (0 \ldots 255),\ i \in \{1, 2, 3, 4\}\}$.

Computing available paths

Different routing protocols use different algorithms to select the shortest path. Routing protocols cannot establish virtual paths without physical connections. Therefore, as the first step, we enumerate all available physical paths in the network. Then according to the configured routing protocol, the specified path selection criterion is used to identify the best paths for the communication. We tested the approach using Rubin's algorithm [34] for enumerating all simple paths[1] in a graph. The efficient implementation encodes the vertices and edges in a path by using bit vectors. The pseudo code of the algorithm appears in Appendix A. This algorithm has N^3 matrix operations.

Definition 3.6 (Path)**.** Path π is a sequence of links and devices along an available physical connection between a source and a destination. Let R_0 be the source, and R_n be the destination of path π, then the k^{th} existing path between R_0 and R_n is defined as follows: $\pi^k_{\langle R_0, R_n \rangle} = R_0 l_1 R_1 \ldots\ldots R_{i-1} l_i R_i \ldots\ldots R_{n-1} l_n R_n$ such that $\forall i, l_i \in L, R_i \in R$ and $NH(R_{i-1}, l_i) = R_i$.

Cost over a path π is $C(\pi) = C(l_1) + C(l_2) + \ldots + C(l_n)$, where $l_1, l_2, \ldots, l_n \in \pi$.

3.2 Filtering Network Model

This section shows the steps of building the filtering network model, which is then used in Chapter 5 for constraint-based analysis. The filtering network model described in this section was published in our paper [35].

[1] A path is called simple if all the vertices of the path are distinct.

To represent a filtering network graph, first, the abstract network graph is taken as the input and the devices are converted into a collection of filtering nodes. This means, a single physical device is split into a number of filtering nodes that describe the effects on the configurations.

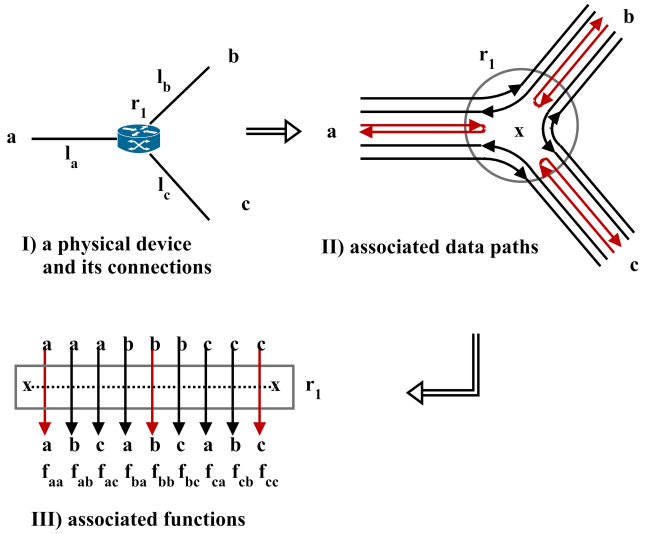


FIGURE 3.3: A device node model

To describe a device by filtering nodes, a *device node model* is defined as shown in Figure 3.3. It interprets a device node as a collection of filters (functions). Each pair of interfaces of a node has assigned a pair of filters and each filter is applied to a single direction. Moreover, there is an internal virtual interface denoted as "x" allowing us to assign a pair of filters that can apply for terminating or originating traffic at the node, respectively.

In graph N_F (see Figure 3.2.c), we can see some interconnecting rules such as two vertices of type V are always connected via a vertex type F, edge E either connects V to F or F to V vertices, vertex type V can have more than two edges connected and vertex type F can only have two edges connected (see Definition 3.7). Because the graph N_F is more complex than graph N_A, graph N_F enables us to model advanced configurations such as policy routing in Cisco devices. Using policy routing, we can enable packets to enter and leave via the same interface, which is called *loopback path*. This will be discussed in the latter part of this section.

Based on the device node model a *filtering network graph* is defined by using a bipartite directed graph as below:

Definition 3.7 (Filtering Network Graph)**.** $N_F = \langle V, F, E_F, \gamma \rangle$, where

- V is a set of vertices representing communication facilities,
- F is a set of vertices representing filtering nodes, i.e. blocks with assigned filters,
- $E_F : V \times F \cup F \times V$ is a set of edges that connect vertices V and F, and
- $\gamma : F \times S \to \mathcal{F}$, where $\mathcal{F}$ is a collection of filters and S is a network state description[2].

For filtering, we mainly consider the Cisco Access Control Lists (ACLs). ACL is an ordered sequence of rules, which permit or deny sending packets from a specific source to specific destination. The following example shows an ACL that permits *http* traffic originating from network 192.168.1.0 by the first rule. Second rule denies the traffic from network 192.168.1.0 to any network, and the last rule permits all traffic. By placing the rule one before the second rule, we can explicitly allow *http* traffic for the network 192.168.1.0. ACL rules are read from top to bottom and first match is the applied rule.

```
permit tcp 192.168.1.0 any www
deny ip 192.168.1.0 any
permit ip any any
```

Formal representation of ACL can be found in [5]. The cascaded ACLs along a path can be combined with "AND" operation and each entry of an ACL will either permit a packet or deny a packet; hence, the result of the filtering functions can be evaluated as Boolean value 1 (permit) or 0 (deny) as shown by Guttman [9]. Therefore, we can define an ACL applied to an interface as below:

Definition 3.8 (Filtering Function)**.** Function $F_{l_{ij}}(p) : ACL \to Boolean$, is the filtering function that evaluates packet p for the Access Control Lists (ACLs) over link l_{ij}.

Filtering function over a path π for the packet p is $F_\pi(p) = F_{l_1}(p) \wedge F_{l_2}(p) \wedge \ldots \wedge F_{l_n}(p)$, where $l_1, l_2, \ldots, l_n \in \pi$. The result of the total filtering function is the conjunction of filtering functions over links along the path π.

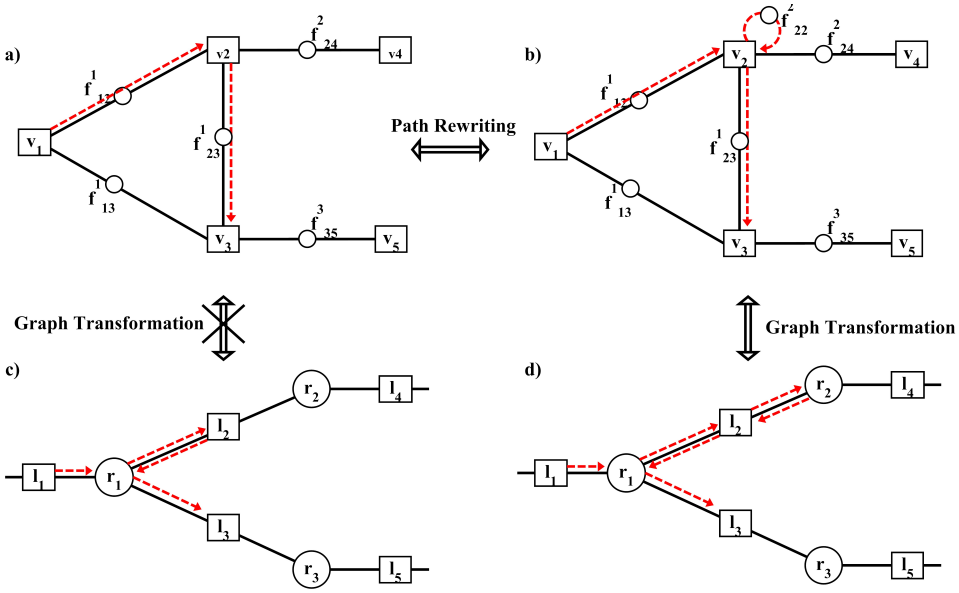
Computing available paths

One may consider a representation of a network by using a filtering network graph as an unnecessary complication. For an example, N_F contains more paths than N_A; these additional paths (spurious paths) are not used in normal configurations. However, this

[2]Depending on the link states in the network, the applied filters will be changed. S is used to map the correct filters from the filtering nodes based on the network state.

type of communication paths is possible to enable with advanced configurations such as policy routing that is able to configure the packets to go through a same device several times.

These spurious paths cannot be directly transformed to the network diagram. If it is required to transform a spurious path, the path should be rewritten before the transformation. Below part explains the path rewriting and the transformation process in detail. Let's consider the paths between vertices l_1 and l_5 in graph N_A and v_1 and v_5 in graph N_F, see Figure 3.4



a) - Filtering network graph
b) - Filtering network graph after path rewriting
c) - Abstract network graph after GT from a)
d) - Abstract network graph after GT from b)

FIGURE 3.4: Converting loopback paths

In N_A there is a single path:

- $\pi_1^A = [l_1, r_1, l_3, r_3, l_5]$

In N_F there are two paths:

- $\pi_1^F = [v_1, f_{13}^1, v_3, f_{35}^3, v_5]$
- $\pi_2^F = [v_1, f_{12}^1, v_2, f_{23}^1, v_3, f_{35}^3, v_5]$

Path $[v_1, f_{12}^1, v_2, f_{23}^1, v_3, f_{35}^3, v_5]$ is spurious. Path $[v_1, f_{13}^1, v_3, f_{35}^3, v_5]$ is real. In the same way, we can show Path $[v_1, f_{12}^1, v_3, f_{23}^1, v_2, f_{24}^2, v_4]$ is spurious. Path $[v_1, f_{12}^1, v_2, f_{24}^2, v_4]$ is real.

Let's consider an example of loopback path between v_1 and v_3, $[v_1, f^1_{12}, v_2, f^1_{23}, v_3]$, which is marked in dotted red line in filtering graph N_F in Figure 3.4.a. The direct transformation will give a non-realistic path $[l_1, r_1, l_2, r_1, l_3]$ in the abstract graph N_A as shown in Figure 3.4.c. Therefore, we need to rewrite the path by using the below *path rewriting rule* as $[v_1, f^1_{12}, v_2, f^2_{22}, v_2, f^1_{23}, v_3]$ in Figure 3.4.b and then convert the rewritten path to the abstract graph N_A as $[l_1, r_1, l_2, r_2, l_2, r_1, l_3]$ in Figure 3.4.d. We can see that there is a loopback path at router r_2, which can be emulated in Cisco routers by configuring policy routing.

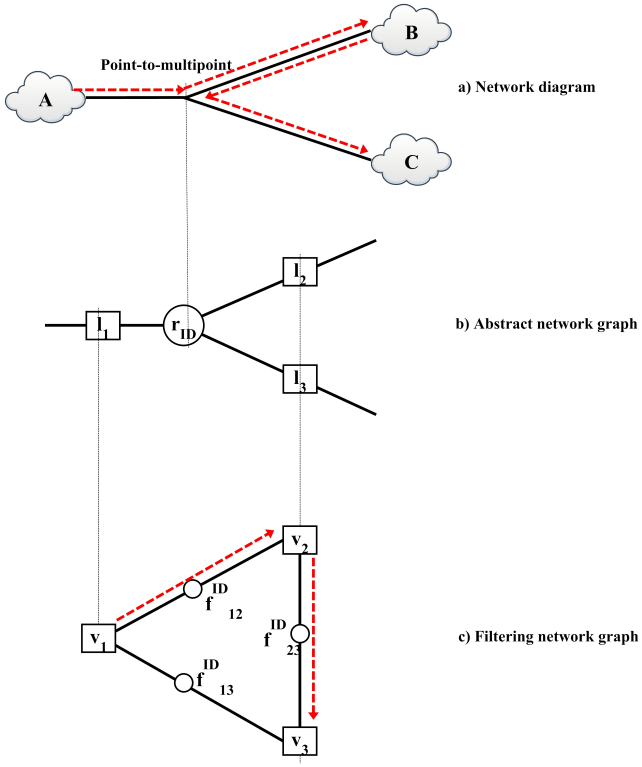


FIGURE 3.5: Point-to-multipoint networks

The loopback path concept can be used to identify and analyze the traffic in point to multipoint connections as shown in Figure 3.5. This type of path analysis is only possible with graph N_F.

Path rewriting rule: If a vertex v_j is in between functional vertices f^p of the same node (r_p) , the vertex v_j should be rewritten as :

$$f^p_{aj} v_j f^p_{jb} \;\Leftrightarrow\; f^p_{aj} v_j f^q_{jj} v_j f^p_{jb} \quad \text{where} \quad r_p \xrightarrow{l_j} r_q$$

This will be used for modeling loopback paths.

The research work presented by Bera et al. [18] proposed the hidden path analysis. This analysis is based on the assumption that there are available services in the network

to allow users to create virtual connections and tunnel data through different types of traffic. The filtering network graph can capture the situation when services are implemented in F-vertices. Therefore, there can be a relation between flow (refer Figure 3.4.a) that starts at v_1 and ends at v_2, and flow that starts at v_2 and ends at v_4. We should check path $[v_1, f_{12}^1, v_2]$ for the first flow and path $[v_2, f_{24}^2, v_5]$ for the second, for instance.

Filtering network graph enables us to find hidden paths. For instance, we can obtain a set of suspicious networks, which can host such services. This means, by analyzing the filters, we can find networks that would allow successful hiding. To identify this type of network threats, the network graph must be able to identify those paths. This is only possible with filtering network graph N_F.

3.2.1 Model Transformation

The filtering network model can be constructed from the abstract network model by performing two steps:

- transforming a network topology in the form of a hypergraph to a filter centering topology described as a bipartite graph, and
- defining assignments of filters to F nodes in the filtering graph according to node configurations in the abstract network model and the given interpretation.

The first step can be formally defined by means of a collection of graph transformation. The graph transformation is detailed in Section 3.3. Second step that requires more attention and detailed implementation of filtering functions based on device configurations, which will be discussed in Chapter 5.

3.3 Graph Transformation

This section introduces graph transformation between graphs N_A and N_F by using Graph Grammar (GG). One advantage of using a formal Graph Transformation (GT) approach is that it has the ability to check the consistency of the properties between graphs. Another advantage is that it can relate the findings during the analysis process to the real network. For an example, if we find a path in graph N_F, then there should be a method to interpret that path in the network diagram in order to physically identify the problematic path in the real network. The presented work consists of basic definitions for GT in algebraic approach based on double pushouts in the Category Theory.

3.3.1 Graph Transformation Definitions

The Graph Transformation Function $\mathcal{T}$ from graph N_A to graph N_F is defined as below:

$\mathcal{T}: N_A \rightarrow N_F$, and $\mathcal{T}^{-1}: N_F \rightarrow N_A$

Graph transformation from the network graph to the abstract network graph is straight forward as below:

- ***Node(r):*** A network device (e.g. router) or links connecting point (switch).
- ***Link(l):*** A connection between adjacent nodes.

We allow ourselves to write $(l_i, r_j, l_k) \in E_A$ for $(l_i, r_j) \in E_A$ and $(r_j, l_k) \in E_A$. Similarly, for E_F we can write $(v_i, f_j, v_k) \in E_F$ if $(v_i, f_j) \in E_F$ and $(f_j, v_k) \in E_F$.

Transformation from abstract network graph to filtering network graph is defined using the following graph transformation mappings:

1. $L \rightarrow V$, which assigns each L-vertex to a unique V-vertex:
   $\mathcal{T}(l_i) = v_i,\ v_i \in V$ and $\forall l_i, l_j$ if $\mathcal{T}(l_i) = \mathcal{T}(l_j)$ then $l_i = l_j$
2. $R \rightarrow 2^F$, which assigns each R-vertex to a set of corresponding F-vertices:
   $\mathcal{T}(r_k) = \{f_{ij}^k \mid \forall i, j\ :\ (l_i, r_k, l_j) \in E_A\}$
3. $E_A \rightarrow F \times V \cup V \times F$, which assigns E_A's to E_F's:
   $\mathcal{T}(l_i, r_k, l_j) = (v_i, f_{ij}^k, v_j),\ (v_i, f_{ij}^k, v_j) \in E_F$ and $(l_i, r_k, l_j) = E_A$

If for all $f \in F$, there exists $r \in R$, such that $f \in \mathcal{T}(r)$ then N_F is the smallest filtering graph that corresponds to network graph N_A by transformation $\mathcal{T}$. We denote this fact as $\mathcal{T}(N_A) = N_F$, and the reverse transformation $\mathcal{T}^{-1}(N_F) = N_A$ can be defined as below:

$\mathcal{T}^{-1}(v_i) = l_i,\ l_i \in L$

$\mathcal{T}^{-1}(f_{ij}^k) = r_k,\ r_k \in R$ and $(v_i, f_{ij}^k, v_j) \in E_F$

$\mathcal{T}^{-1}(v_i, f_{ij}^k, v_j) = (l_i, r_k, l_j),\ (l_i, r_k, l_j) \in E_A$ and $(v_i, f_{ij}^k, v_j) \in E_F$

Properties of transformation function

It is possible to extend $\mathcal{T}$ for paths. Then, the transformation $\mathcal{T}$ preserves the following path properties:

- If $\exists\pi$ in N_A: $\exists\pi^{'}$ in N_F such that $\mathcal{T}(\pi) = \pi^{'}$
- If $\exists l_i \in \pi$ in N_A: $\exists v_i \in \pi^{'}$ in N_F such that $\mathcal{T}(l_i) = v_i$
- If $\exists l_i r_k l_j \in \pi$ in N_A: $\exists f_{ij}^k \in \pi^{'}$ in N_F such that $f_{ij}^k \in \mathcal{T}(r_k)$
- If $r_k l_i r_m l_i r_k$ is a *loopback path* in N_A, then $\exists f_{xi}^k v_i f_{iy}^k$ in N_F such that $f_{xi}^k v_i f_{iy}^k \Rightarrow f_{xi}^k v_i f_{ii}^m v_i f_{iy}^k$ (from path rewriting rule), and $\mathcal{T}^{-1}(f_{xi}^k v_i f_{ii}^m v_i f_{iy}^k) = l_x r_k l_i r_m l_i r_k l_y$ (which will be a loopback path)

3.3.2 Graph Grammar

This subsection introduces a method for GT by using GG described in category theory.

First part of this subsection presents the basic definitions. These definitions involve GT in classical algebraic approach, based on the double pushouts in the category of **Graphs**. Second part describes on the use of GG for network GTs supported by a detailed example.

Graph transformation with graph grammar

There are five basic definitions involved in the process of GT by using GG. First definition defines graph by using the category theory and second defines the typed graph. Then we need to define the graph productions that are the basic elements of the GT and the process of GT by using the defined graph productions. Last definition shows how to implement a GT system by using graph language.

More details on GT and its properties can be found in the second and third chapters of the book by Ehrig et al. [36].

Definition 3.9 (Graph)**.** A graph production $G = (V, E, s, t)$ consists of a set V of nodes (also called vertices), a set E of edges, and two functions $s, t : E \longrightarrow V$, the source and the target functions.

Definition 3.10 (Typed Graph)**.** This is a distinguished graph $TG = (V_{TG}, E_{TG}, s_{TG}, t_{TG})$. V_{TG} and E_{TG} are called the vertex and the edge type alphabets, respectively.

A tuple (G, type) of a graph G together with a graph morphism type : $G \longrightarrow TG$ is called a typed graph.

Definition 3.11 (Graph Production)**.** A (typed) graph $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ consists of (typed) graphs L, K and R, called the left-hand side graph, gluing graph, and right-hand side graph respectively, and two injective (typed) graph morphisms l and r.

Given a (typed) graph production p, the inverse production is defined by
$p^{-1} = (R \overset{r}{\leftarrow} K \overset{l}{\rightarrow} L)$

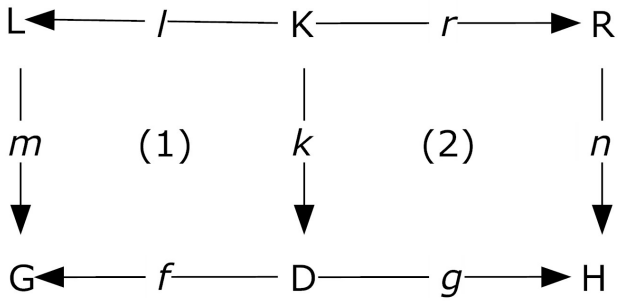


FIGURE 3.6: Graph transformation

Definition 3.12 (Graph Transformation)**.** Given a (typed) graph production $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ and a (typed) graph G with a (typed) graph morphism $m : L \longrightarrow G$ called the match, a direct (typed) graph transformation $G \overset{p,m}{\Longrightarrow} H$ from G to a (typed) graph H is given by the double-pushout (DPO) as shown in Figure 3.6, where (1) and (2) are pushouts in the category of (typed) Graphs.

Inverse graph transformation: Given a direct (typed) graph transformation $G \overset{p,m}{\Longrightarrow} H$ with a matched morphism $n : R \longrightarrow H$ then there is a direct (typed) graph transformation $H \overset{p^{-1},n}{\Longrightarrow} G$

Definition 3.13 (GT System, Graph Grammar, and Language)**.** A graph transformation system GTS = (P) consists of a set of graph productions P.

A typed graph transformation system GTS = (TG,P) consists of a type graph TG and a set of typed graph productions P.

A (typed) graph grammar GG = (GTS, S) consists of a (typed) graph transformation system GTS and a (typed) start graph S.

The (typed) graph language $\mathcal{L}$ of GG is defined by

$\mathcal{L} = \{G \mid \exists\ (typed)\ graph\ transformation\ S \overset{*}{\Longrightarrow} G\}$

Graph Grammar for network transformation

This explains GT between our network graphs, abstract network graph and filtering network graph by using GG. First part defines the basic production rules and second part shows an application of GT by using an example. Since the l vertices in N_A are same as v vertices in N_F, to reduce the complexity of the figure, we have denoted both vertices by v in our example.

Graph Productions

This GT can be performed by defining three Graph Productions (GP). $p = (L \xleftarrow{l} K \xrightarrow{r} R)$. The three production rules are named as *expansion rule*, *full-mesh rule* and *termination rule*, and denoted as p_1, p_2 and p_3 respectively.

I. Expansion Rule

Production p_1 is graphically shown in Figure 3.7. The main objective of this production is to group the connected links of one network device. The vertex (R) represented by a circle expands along each edge till it meets the square vertices (V). Since the number of connecting edges to a vertex can vary, the graphical representation shows a general instance of a circle vertex connected with v_1 to v_i square vertices through edges.

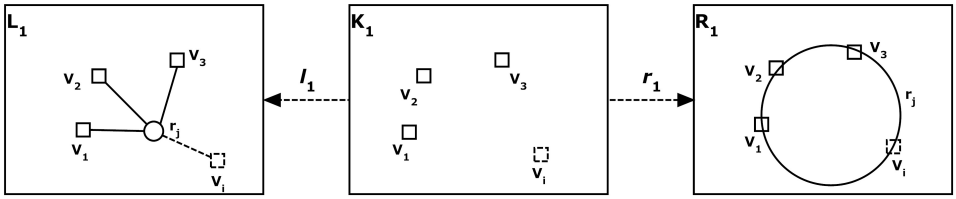


FIGURE 3.7: Expansion rule, $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1)$

II. Full-Mesh rule

Once the related square vertices are grouped by the above rule, the filtering nodes should be introduced. Therefore, all the grouped square vertices are connected in the manner of full-mesh with filtering vertices by using the production p_2 as graphically shown in Figure 3.8. For illustration purposes the introduced F-vertices are shown in black circles.

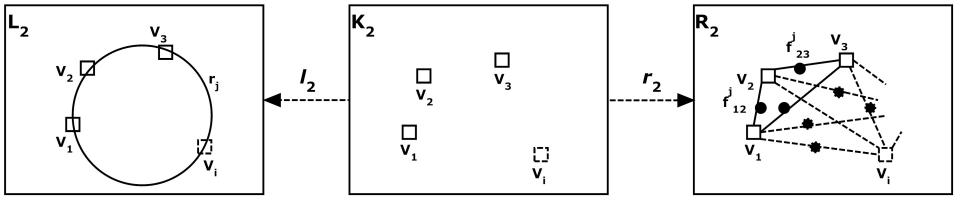


FIGURE 3.8: Full-Mesh Rule, $p_2 = (L_2 \overset{l_2}{\leftarrow} K_2 \overset{r_2}{\rightarrow} R_2)$

III. Termination Rule

This is a special instance of the Full-Mesh rule, which is the expanded circle is having only one square vertex. In this case, the square vertex will be connected to a filtering node with virtual interface by using production p_3 as graphically shown in Figure 3.9. This is applicable only for the terminating nodes.

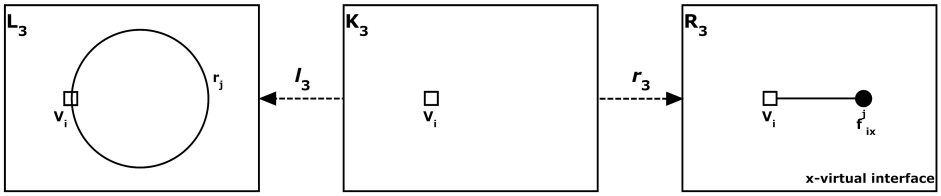


FIGURE 3.9: Termination rule, $p_3 = (L_3 \overset{l_3}{\leftarrow} K_3 \overset{r_3}{\rightarrow} R_3)$

Graph Transformation

This part shows the network GT by using GG. For the GT, we will use the three graph productions defined in the previous part. The abstract network graph G_1 in Figure 3.10 is taken as an example to illustrate the graph transformation. The final graph G_2 will be the filtering network graph of G_1. To visually relate the transformation, the positions of square vertices have been kept unchanged.

The below illustrated GT example has two stages. GT for the first set of nodes is shown in Figure 3.10, and the second set of nodes is shown in Figure 3.11. During the GT process, one graph production rule can be applied for all applicable nodes at once. For an example, expansion rule can be applied for r_0, r_1, r_2, r_3, r_5 and r_8 nodes at once. By performing the GT in two steps, we are trying to show that the GT is closed under parallel and sequential transformations. The GT from G_1 to G_2 can be explained in detail as below.

The complete transformation is performed in five steps as $G_1 \overset{p_1,m_1}{\Longrightarrow} H_1 \overset{p_2,m_2}{\Longrightarrow} H_2 \overset{p_1,m_3}{\Longrightarrow} H_3 \overset{p_2,m_4}{\Longrightarrow} H_4 \overset{p_3,m_5}{\Longrightarrow} G_2$.

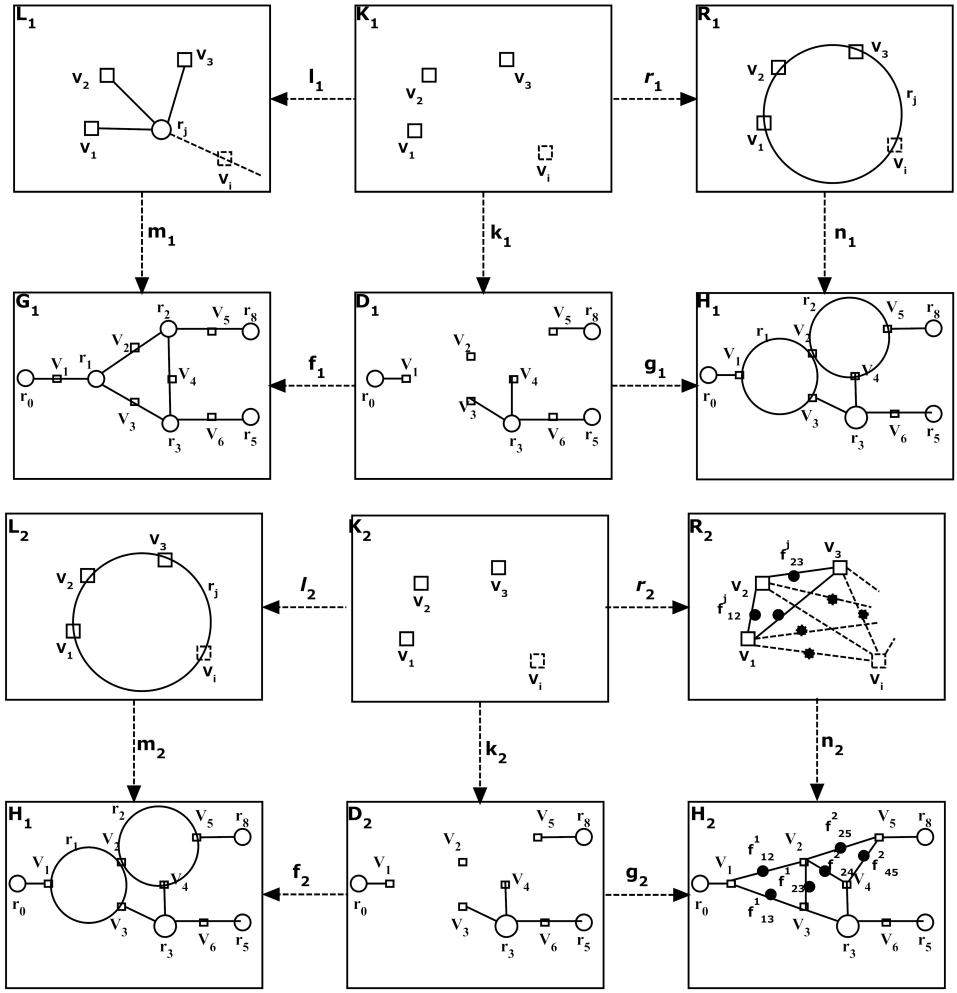


FIGURE 3.10: An example of GT Part-1

First and the second steps of GT are shown in Figure 3.10. First step of GT is applied for the nodes r_1 and r_2 with the GP expansion rule. This will produce the graph H_1 where $G_1 \overset{p_1,m_1}{\Longrightarrow} H_1$. The second step of GT is applied for graph H_1 with the GP full-mesh rule, this will produce graph H_2 where $H_1 \overset{p_2,m_2}{\Longrightarrow} H_2$.

Then the third, fourth and the fifth steps of GT are shown in Figure 3.11. In graph H_2, the nodes r_0, r_3, r_8, r_5 are taken for GT with expansion rule that then produce the graph H_3, further, GT is used with full-mesh rule to generate the functional vertices that form the graph H_4 where $H_2 \overset{p_1,m_3}{\Longrightarrow} H_3$ and $H_3 \overset{p_2,m_4}{\Longrightarrow} H_4$.

Even though we depict the GT in two separate figures, the processing of the r nodes can be done at once. The GT was performed in this way to show that transforming nodes in parallel or sequentially will not affect the final graph. For an example, the transformation $G_1 \overset{p_1,m_1}{\Longrightarrow} H_1$ *and* $H_1 \overset{p_2,m2}{\Longrightarrow} H_2$, the vertices r_1, r_2 were matched to show the parallel transformation. For $H_2 \overset{p_1,m_3}{\Longrightarrow} H_3$, vertices r_0, r_1, r_2, r_3, r_5 were matched for

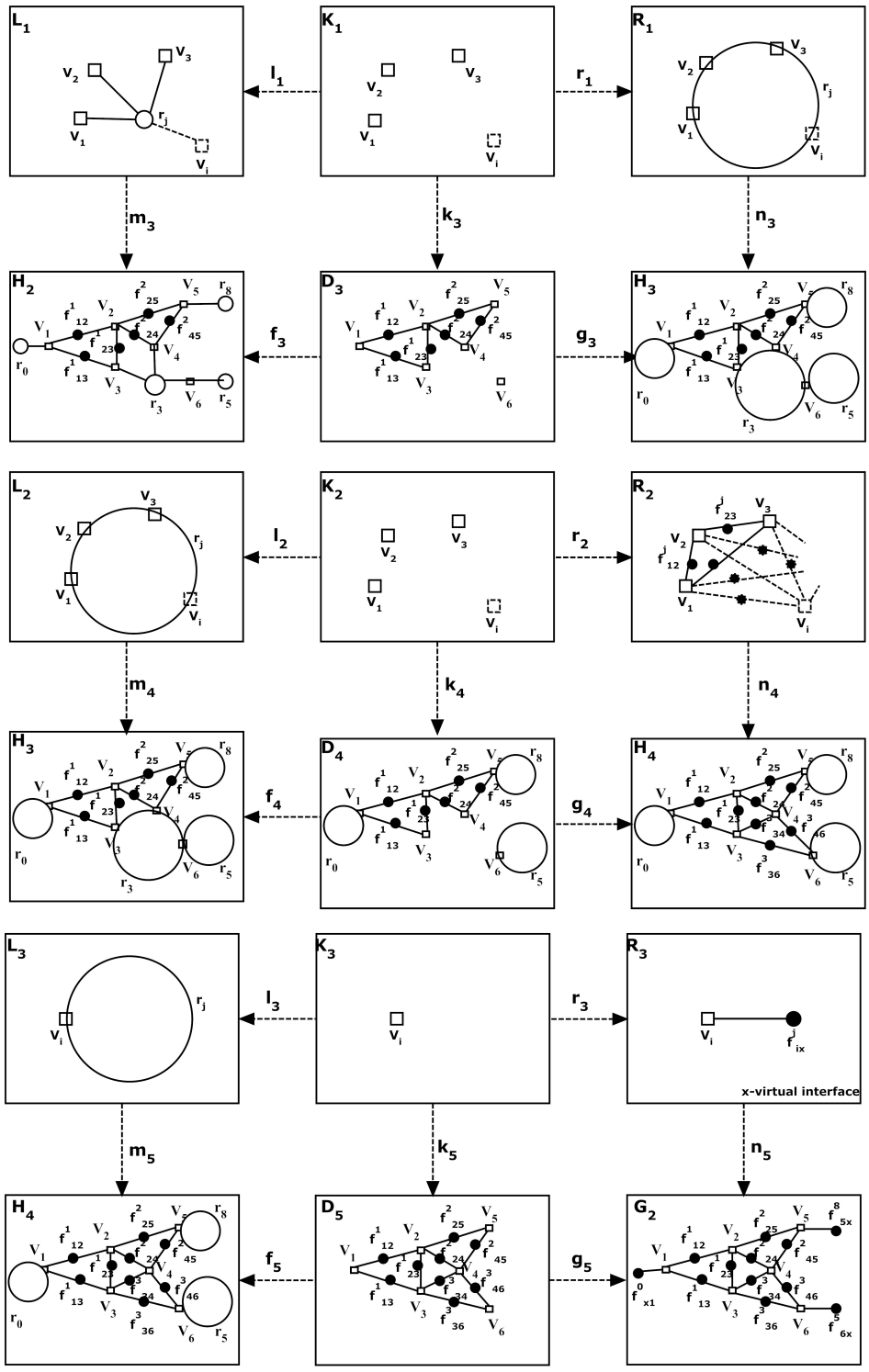


FIGURE 3.11: An example of GT Part-2

transformation in the second stage to show the sequential transformation with nodes r_1 and r_2.

The last part of the GT process is to transform the termination nodes, which is $H_4 \overset{p_3,m_5}{\Longrightarrow} G_2$. One advantage of this GT is that the sequences of selected nodes are invariant for the final graph. Even though the underlying theory of GT involves high level of algebraic representation and requires deeper knowledge of category theory to work with formulas, the network GT can be performed with three simple rules. Therefore, the network GT can be automated easily to transform abstract network graph N_A to filtering network graph N_F.

3.4 Model of Forwarding Devices and Routing Information

This section briefly defines a different network model based on a combination of the abstract network graph and routing information base. A path-based approach is used for modeling, which is for each pair of source and destination a set of paths that satisfy a certain criteria is determined. These paths are found in the graph of a network topology that captures physical interconnections among network devices and connected networks. Each protocol maintains its knowledge in a structure called Routing Information Base (RIB). Routers collect information from individual RIBs and maintain Forwarding Information Base (FIB), which governs the forwarding of packets. The work described in this section was published in our paper [37].

First part of this section highlights different types of RIBs and the information flow between them. Then we will show how to used RIBs and FIBs together with network topology to represent routing information. This method is used for routing analysis, which will be detailed in Chapter 6.

- *Local RIB* is stored in the routing process running on a router. Each process has its own RIB, e.g. RIP maintains RIP database. Similarly, *local FIB* is a single data table, which is used by a router to decide where to forward the incoming packets.
- *Network RIB/FIB* is a network wide view of routing information. This represents a shared routing knowledge of forwarding devices. Similarly, network FIB represents a global view of routing information that governs forwarding packets in the entire network.
- *Forwarding device* is a network device that actively decides where to forward the packets based on its local routing information stored in FIB.

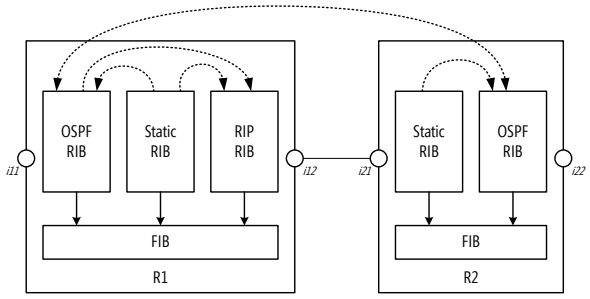


FIGURE 3.12: An example of router model

- *Redistribution* stands for copying routing information to a target protocol instance, which is done in the scope of a single router.
- *Routing Protocol* defines rules of routing information exchange and routing information synthesis on the local router. Commonly, Routing Information Protocol (RIP) , Open Shortest Path First (OSPF), Interior Gateway Routing Protocol (IGRP), and Enhanced IGRP (EIGRP) are employed in local networks.
- *Routing Instance* is also called as routing process. This is a process that runs the implementation of routing protocol within router's boundaries. It interacts with routing instances running at neighboring routers.

Next section describes the behavior of the forwarding devices and the process of modeling them. This model reveals an abstract internal structure of a router with respect to routing instances.

3.4.1 Modeling Forwarding Devices

A packet forwarding device (router) is modeled as a collection of routing instances where each maintains its own routing information base (RIB). Router forwards packets referring to the forwarding information base (FIB). The FIB is populated from local RIBs according to a specified procedure, which is usually proprietary of each device vendor. To build the model, in this section, Cisco devices are considered as the reference platform. On this platform each routing protocol is assigned an administrative distance (see Table 4.2), which specifies the priority of the information stored in RIB with respect to router's FIB. Routing protocols with lower administrative distances are maintaining more accurate routing information, and hence their information is equipped by a higher priority than information of routing protocols with higher administrative distances.

Figure 3.12 shows an example of a model of a router. This model is essentially the same as defined by Maltz et al. [38] and the rest of this subsection will restate the key features

of this model including graph of routing information flow denoted as G_{RIB}. In Figure 3.12, there are three routing processes, denoted as Static [3], OSPF, and RIP. The arrows represent the following information flows:

- A flow from RIB to FIB represents the process of populating FIB with selected items from RIB according to the defined rules, e.g. based on administrative distance.
- A flow between RIBs represents a *redistribution* of information between different routing protocols or different instances of the same routing protocol running on the same router.
- A flow between RIBs on different devices represents information sharing (or exchanging) between routers that are using the same routing protocol.

The graph G_{RIB} can be defined based on the above information. This information can be gathered from network configuration files. Routing information flows form a graph $G_{\mathsf{RIB}} = \langle V_{\mathsf{RIB}}, E_{\mathsf{RIB}}, \mathcal{P} \rangle$ where V_{RIB} is the set of RIBs in the network and E_{RIB} is the set of adjacencies between RIBs over which routing information can flow. Set $\mathcal{P}$ contains properties that can be assigned to edges E_{RIB}. This model is used for the routing analysis described in Chapter 6.

3.4.2 Cost Function for RIP, OSPF and EIGRP

This subsection shows the methods used by different routing protocols to compute the costs. These calculated costs are used by FIB to select the best route from the collection of RIBs within the router.

- *RIP* [39] – RIP has default link cost of value one, and hence for any link l, $C(l) = 1$. Therefore, the shortest (best) path has the lowest hop count.
- *OSFP* [40] – OSPF requires the bandwidth function (BW) to compute the cost function, $BW(l_i) : L \to \mathcal{N}$, where $BW(l)$ is the bandwidth of the link l. The cost function for a link is defined by Cisco [41] as $C(l) = [10^8/BW(l)]$. The cost of the path $C(\pi)$ has the accumulated link costs along the path and the least cost path will be used for the communication.

[3] A Static routing process maintains the configured static information on a router. It consists of directly connected networks and static routes, which are inserted by administrators.

- *EIGRP* [42] – EIGRP uses delays and bandwidths of the whole path to calculate the cost function of the path. Delay function D is $D(l_i) : L \to \mathcal{N}$. Delay function for the path is defines as $D(\pi) = D(l_0)+D(l_1)+\ldots+D(l_n)$, where $l_0, l_1, \ldots, l_n \in \pi$. Bandwidth function BW is $BW(l_i) : L \to \mathcal{N}$. Bandwidth function for the path is defined as $BW(\pi) = BW(l_i)$ where $\forall l_i, l_j \in \pi, BW(l_i) \leq BW(l_j)$. Then the cost function for the path π is given by $C(\pi) = [10^8/BW(\pi) + D(\pi)]$.

3.4.3 Representing Routing Information

A router's FIB stores routing information in a form of a record consisting an identification of a destination network. The next hop router, which is either determined by specifying an outgoing interface or by its IP address, and a cost. In this path based model, a global view of a network is considered. The network FIB is represented by FIB matrix and it contains best paths among all destinations. Each cell of this matrix contains information in the form of $r_1 \overset{c_1}{\to} r_2 \overset{c_2}{\to} \ldots r_{n-1} \overset{c_{n-1}}{\to} r_n$, where $r_i \in R$. Alternatively, this can be written as $\langle \pi, c \rangle$, where $\pi = r_1, \ldots, r_n$ is a path and c is an aggregate cost. A cost of the path is always interpreted with respect to an advertised routing protocol (see Section 3.4.2). Therefore, it requires to annotate every costs with its meaning, i.e. $c_i = \langle p, q \rangle$, where p denotes a path segment and q interprets that segment cost .

A *cost-path* is a tuple $\langle \pi, c \rangle$, where π denotes a path and c is an aggregate costs to the destination. A path may contain sub paths, for instance, $\langle (\langle (r_1, r_2, r_3), 2 \rangle, r_6, r_5, r_8), 5 \rangle$. This allows us to model a path that is observed by employing multiple protocols using redistribution.

The objective is to express a global view of the network routing information. Hence, instead of modeling local RIBs and FIBs, two other information based are defined, they are namely the network RIBs and the network FIB. There are many network RIBs where the number depends on the number of routing instances running on the network. There is always at least one network RIB that models the static routing. There is always a single network FIB, which contains complete information on routing for the current network state.

The detailed building of the network RIBs from the configuration files and the reachability analysis process are presented in Chapter 6.

3.4.4 Forwarding State

A forwarding state describes a condition of a network determined by a content of network forwarding information base (NFIB). In general, the content of NFIB for a given network

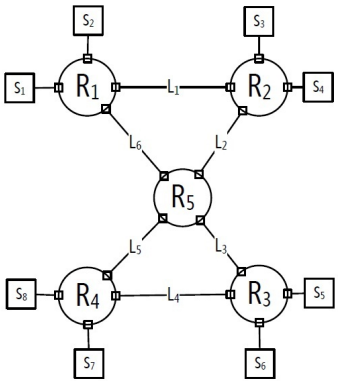


FIGURE 3.13: An example of network topology

$\mathcal{F}$	*Map*	$\mathcal{F}$	*Map*	$\mathcal{F}$	*Map*
f_{12}^{L}	$L_1 \rightarrow L_2$	f_{12}^{S}	$S_1 \rightarrow S_2$	f_{11}^{SL}	$S_1 \rightarrow L_1$
f_{21}^{L}	$L_2 \rightarrow L_1$	f_{21}^{S}	$S_2 \rightarrow S_1$	f_{11}^{LS}	$L_1 \rightarrow S_1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
f_{54}^{L}	$L_5 \rightarrow L_4$	f_{87}^{S}	$S_8 \rightarrow S_7$	f_{58}^{SL}	$L_5 \rightarrow S_8$
f_{45}^{L}	$L_4 \rightarrow L_5$	f_{78}^{S}	$S_7 \rightarrow S_8$	f_{85}^{LS}	$S_8 \rightarrow L_5$

TABLE 3.1: Filter map

depends on the health of devices, state of links, and external routing information pumped into the network.

Consider the network in Figure 3.13 where all routers are running RIP and exchange all information on connected sub networks. The available filter maps are shown in Table 3.1. This shows the filters maps within a device. When the device interface is connected to a link notation L is used; and when the device interface is connected to a service network, the notation S is used. RIP uses hop count as the metric. The shortest path from S_1 to S_6 is $\langle S_1, R_1, L_6, R_5, L_3, R_3, S_6 \rangle$ when all links are up. This path is placed in NFIB. If links L_6 and L_3 are unavailable then NFIB would contain longer path $\langle S_1, R_1, L_1, R_2, L_2, R_5, L_5, R_4, L_4, R_3, S_6 \rangle$. Similarly, NFIB can be defined for other paths. Table 3.2 shows a format of NFIB for two forwarding states q and r.

It is possible that different combinations of link states or device failures can lead to the same forwarding state, since links contain different topological importance for packet forwarding. The assessment of links and routers on the importance of traffic delivery is detailed in our paper [32].

q	S_1	...	S_6	...
S_1	$\cdot$	...	R_1, L_6, R_5, L_3, R_3	...
⋮	⋮	⋮	⋮	⋮

r	S_1	...	S_6	...
S_1	$\cdot$	...	$R_1, L_1, R_2, L_2, R_5, L_5, R_4, L_4, R_3$	...
⋮	⋮	⋮	⋮	⋮

TABLE 3.2: NFIB for state q and r

3.5 Discussion

Network modeling is the most important part of this thesis. Based on the correctness and the assumptions used to build the network model, the accuracy of the results of the analysis can vary. Another important fact is the expressiveness of the models. This refers, how far a model can describe network properties and behaviors. For an example, the abstract network model cannot identify the loopback paths and the filtering network model can identify the loopback paths. Another important point is the computational cost of the models. Since we are planning to automate the models by reading device configurations on-line, the models can be built within a less time frame without any human interaction.

Even though each model has its own advantages and disadvantages over the other, we choose the relevant model based on the analysis that we intend to perform. For an example, constraint-based analysis in Chapter 5 requires a model that can represent detail device configurations; therefore, we have used the filtering network model. Reachability analysis described in Chapter 4 only requires representation of the correct topology; therefore, we have used the abstract network model. If we have used abstract network model for routing analysis described in Chapter 6, then we would need to build routing tables, and interpret routing filters and routing updates. This would be a difficult task; therefore, from the analysis of routing, we have used our third model based on the RIBs.

When we compare our models with the existing network models, the direct convention from network topology to a graph model is the most widely used approach. The filtering network model is a new model, which we have introduced. The last model of forwarding device is an extension of the previous work (See Maltz et al. [38]).

In summary, this chapter detailed on network modeling methods by using graph theory. Three models were discussed. First model was built by using formal modeling and named as abstract network model, which will be used for formal analysis. The second model

was built by using functional abstractions of device node model and called the filtering network model, which will be used for constraint-based analysis. The last model consists of abstract network graph together with the routing information base, which will be used for routing analysis. Figure 1.2 shows an overview of the interconnections between the network models and analysis methods described in this thesis. The following chapters will discuss on different types of analysis methods by using these three network models.