2011 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems

Reachability Analysis in Dynamically Routed Networks

Miroslav Sveda Ondrej Rysavy Gayan de Silva
Petr Matousek Jaroslav Rab
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
Email: {sveda,rysavy,xdesil,matousp,rabj@fit.vutbr.cz}

***Abstract*—In this paper, we introduce a novel approach to reachability analysis of dynamically routed networks. The goal is to determine the network-wide reachability using static analysis of configuration files gathered from forwarding devices. We describe a method that can compute the reachability in networks with a mix of static routing configurations, distance vector routing protocols, filtering routing updates and redistributions. The method computes a network-wide approximation of distributed routing information using the standard graph algorithms. Thus, for any network state, we can determine a set of active paths used for packet delivery. The outcomes of the method can be, for instance, used during the conformance checking of distributed access control lists against network security policies.**

***Keywords*-IP-networks; network configuration; network design; network reachability; routing protocols**

I. Introduction

Network design belongs to complex tasks. Network specialists should fulfill customers requirements while considering the limits of underlined technologies. The goal is to provide reliable network services as requested. Once the design is finished, the deployment phase is launched. It consists of installation and physical interconnection of the devices, setting up their configurations and, finally, network troubleshooting, in order to assure network functionality. Identification of potential problems as early as possible in the design phases appears a serious argument for extra techniques and methodologies that verify and validate the results of the design process.

Routing design can be very complex task in practice as contemporary converged networks have to achieve many different objectives, e.g., basic reachability, network security, quality of services required by converged applications, resilience, or scalability. A lot of implementation options are available to meet the design goals where each of them imposes certain constraints and side-effects. To validate routing designs, the network is usually set up and troubleshot in an laboratory environment to find the acceptable design and to correct possible misconfigurations.

In this paper, we focus on the static analysis of network configuration files, which may be helpful in network design and configuration. In this sense, we share the view of Feamster [1] who explicitly points out the similarity of Border Gateway Protocol (BGP) configuration verification to program analysis problem. We also consider a simple failure model that helps network designers to check if the configuration can guarantee availability of critical network services under a presence of eventual device or link failures. Of course, the method presented in this paper cannot guarantee the resilience of a network, but it can reveal misconfigurations that may cause to happen unwanted network scenarios.

While most of the related work (except [2]) ommit the issue and effect of routing, we attempt to create a model that primarily considers it. Routing issue may be ignored in case of verifying Access Control List (ACL) implementations, which should be consistent even in the case of unpredictable content of routing tables, but cannot be neglected in the case of quality of service analysis, where properly configured dynamic routing enables to achieve quality requirements of network services.

A. State of the Art

The analysis of routing design including packet filtering and routing policies recently attracks interest among researchers because of demands to build larger and more capable networks. In this subsection, we shortly survey research done in security policy verification, packet filters validation and routing design analysis.

In 1997, Guttman defined a formal method to compute a set of filters for individual devices given a global security policy [3]. To achieve a feasibility, the network is abstracted so only network areas and border routers occur in a model. This natural decision mirrors the real situation as internal routers do not usually participate in data filtering. Similarly, data flow model is defined in terms of abstract packets, which are described by abstract source and destination addresses and service types. An algorithm computes a feasibility set of packets that passes all filtering rules along the path. The employed abstract packets description make the procedure practically feassible and efficient.

Yan et al. developed a tool called FIREMAN [4], which allows to detect misconfigurations in firewall configurations. The FIREMAN performs symbolic model checking of the firewall configurations for all possible IP packets and along

978-0-7695-4379-6/11 $26.00 © 2011 IEEE
DOI 10.1109/ECBS.2011.24

all possible data paths. The underlaying implementation depends on Binary Decision Diagram (BDD) library, which efficiently models firewall rules. This tool can reveal intra-firewall inconsistencies as well as misconfigurations that lead to errors at inter-firewall level. The tool can analyze ACL series on all paths for end-to-end connections thus offering network-wide firewall security verification.

Jeffrey and Samak in [5] aim at firewall configurations analysis using bounded model checking approach. They focus 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 firewall configuration, it means to find a packet p, which is not matched by any rule of the firewall set. They implemented an analysis algorithm by translating the problem to a SAT instance and showed that this approach is efficient and comparable to tools based on BDD representation. Similar work was done by Pozo, Ceballos and Gasca [6] who provided a consistency checking algorithm that can reveal four consistency problems called shadowing, generalization, correlation and independency.

Liu et al. developed a method for formal verification and testing of (distributed) firewall rules (see [7], [8], [9]) to the user provided properties. They represent firewall rules using a Firewall Decision Diagram (FDD), which forms an input to a verification algorithm. Another input is a property rule, which describes a property that thay want to check, e.g. description of a set of packets that should pass the firewall. By a single traversing an FDD structure from the root to the leave it is possible to check the given property. In [2], Xie et al. report on their extensive work on static analysis of IP networks. They define a framework able to determine lower and upper approximations on network rechability considering filtering rules, dynamic routing and packet transformation. The method computes a set of packets that can be carried by each link and, thus, by combination of this, it is possible to determine 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 the paper, the authors also present refinement of both upper and lower approximations by considering effect of routing.

Bera, Dasgupta and Ghosh (see [10], and [11]) define a verification framework for filtering rules that allows one to check the correctness of distributed ACL implementations against the given global security policy and also to check reliability (or fault tolerance) of service in a network. To check the correctness, the filtering rules are translated to assertions represented as boolean formulas that are together with translation of the global security policy sent to SAT solver. In the case of inconsistency the SAT solver may produce a counter example that helps administrator to debug ACL rules. To check the reliability, the framework accepts a description of a global security policy, a collection of ACL rules and a network description to compute whether the rules are consistent with the given policy. A policy is understood as a description of service availability with respect to defined network zones. First, the method computes a network access model, which is a directed graph with ACLs assigned to its edges. Next, the Service Flow Graphs (SFG) are generated for all service in the interest, for instance, SFG for ssh traffic. An SFG is a subgraph of network accesss graph. Based on this graph the fault analysis is performed by computing values of minimum cut in all SFGs. These values then represent how many link failures can be tolerated.

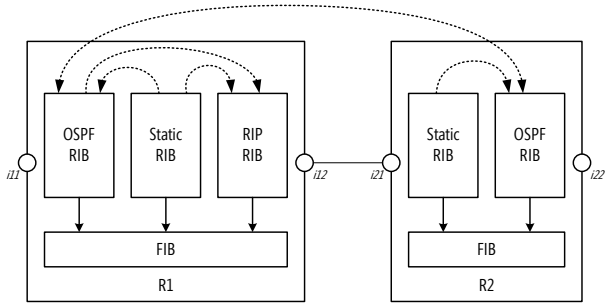


Figure 1. A router model

B. Contribution

The contribution of this paper consists in the development of a novel method focused on checking end-to-end connectivity in dynamically routed networks[1]. Given configurations of forwarding devices and a network topology, the method is able to determine feasible paths in the network that consequently allows us to check the overall reachability of network services. The outcomes of the method can be further deployed in analysis of the security level implemented by firewalls distributed along the multiple paths. Using this analysis, it is, e.g., possible to find backdoors or hidden paths [10] in a network that can be used for the unauthorized access to network services.

C. Structure of the paper

In section II, we restate a formal model for networks, network nodes (routers) and routing information. In section III, we define a method for computing all paths in a network with only static routing information available. This method is then extended in section IV to cope with distance vector routing protocols. In particular, we focus on the issue of filtering routing updates. Section V and VI deals with the route redistribution and the selection of routing information from routing information bases of routing protocol instances to network-wide forwarding information base that gives the desired output: the collection of all active paths in the

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

network. We conclude the paper in section VII discussing the limitations of the method and the further work.

II. Modeling Forwarding Devices and Routing Information

In this section we give a definition for network model and a description of routing information. We use a path-based model, which for each pair of source and destination networks determines a set of paths that satisfy a certain criteria. 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 structures called Routing Information Base (RIB). Routers collect information from individual RIBs and maintain Forwarding Information Base (FIB), which governs the forwarding of packets. Next, we state the basic terminology used in this paper.

A. Terminology

Thorough the paper, we commonly refer to the following terms:

- *Local RIB* is stored in routing process address space running on a router. Each process has its own RIB, e.g. RIP maintains RIP database. Similarly, *local FIB* is a single datatable, which is used by a router to decide where to forward 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 on routing information that governs forwarding packets in the entire network.
- *Forwarding device* is a network device that actively decides where to forward packets based on its local routing information stored in FIB.
- *Redistribution* stands for copying route 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 at 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* also called routing process is a process that runs the implementation of routing protocol within router's boundaries. It interacts with routing instances running at neighboring routers.

B. A Network Model

We model the network topology as a hypergraph $G_{\mathsf{NET}} = \langle V_{\mathsf{NET}}, E_{\mathsf{NET}}, \mathcal{C} \rangle$, where V_{NET} is a set of forwarding devices (routers) and $E_{\mathsf{NET}} \subseteq 2^{V_{\mathsf{NET}}}$ is a set of physical links [2], and $\mathcal{C}$ is a set of configurations that govern behaviors of forwarding devices. For any $v \in V_{\mathsf{NET}}$ there is a configuration $C_v \in \mathcal{C}$. More information about a similar network model can be found in our previous paper [12].

Similarly, we define a graph for each routing instance in a network. Before we can do that, we need to introduce a model of forwarding devices. This model reveals an abstract internal structure of a router with respect to routing instances.

C. The Model of Forwarding Device

A packet forwarding device (router) is modeled as a collection of routing instances each maintaining its own routing information base (RIB). Router forwards packets using information from forwarding information base (FIB). The FIB is populated from local RIBs according to specified procedure, which is usually proprietary to each device vendor. In this paper, we use Cisco devices as the reference platform. On this platform each routing protocol is assigned by administrative distance that specifies a priority of the information stored in RIB with respect to router's FIB. Routing protocols with lower administrative distances are believed to maintain more accurate routing information and hence their information is equipped by a higher priority than information of routing protocols with higher administrative distances.

Figure 1 depicts the model of a router. This model is essentially the same as defined by Maltz et al. in [13] and in the rest of this subsection we restate the key features of this model including graph of routing information flow denoted as G_{RIB} (see bellow). In Figure 1, 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.

Based on the previous information which all can be gathered from configuration files we can define a graph G_{RIB}. 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}. In section IV, $\mathcal{P}$ is interpreted as a set of route filters.

[2] We model a network as hypergraph because we also admit topologies that include n-to-n connections.

[3] The Static routing process maintains static information configured on a router. It consists of directly connected networks and static routes inserted by an administrator.

D. Representing Routing Information

A router's FIB stores routing information in form of a record consisting of identification of a destination network, a next hop router, which is either determined by specifying an outgoing interface or by its IP address, and a cost. In our path based model, we keep a global view on a network. Thus we determine network FIB, which is represented by FIB matrix that contains best paths among all destinations. Each cell of this matrix contains information in the form of $v_1 \rightarrow^{c_1} \rightarrow \ldots v_{n-1} \rightarrow^{c_{n-1}} v_n$, where $v_i \in V_{\mathsf{NET}}$. Alternatively, we can write $\langle \pi, c \rangle$, where $\pi = v_1, \ldots, v_n$ is a path and c is an aggregate cost if we do not care about particular link costs. A cost of the path is always to be interpreted with respect to a routing protocol that advertises such path, e.g. RIP uses hop-count while OSPF uses values proportional to bandwith along the path. Thus we need to annotate every costs with its meaning, i.e. $c_i = \langle p, v \rangle$, where p denotes an interpretation for cost v.

A *cost-path* is a tuple $\langle \pi, c \rangle$, where π denotes a path and c is an agregate costs to the destination. A path may contain subpaths, for instance, $\langle \langle (v_1, v_2, v_3), 2 \rangle, v_6, v_5, v_8), 5 \rangle$. This allows us to model a path that was observed by employing multiple protocols using redistribution.

Our aim is to express a global view on the network routing information; hence, instead of modeling local RIBs and FIBs, we define a network RIBs and the network FIB. There are many network RIBs, where the number depends on the number of routing instances running in network. There is always at least one network RIB that models the static routing. There is always a single network FIB, which contains a complete information on routing in the current network state. In the following sections we show how network RIBs and the network FIB are computed from static routing configurations.

III. Static RIBs

Static routing information base (RIB) for every router is obtained from analysis of router configuration files. For instance, in the case of Cisco devices the static information appearing in the routing table consists of directly connected networks and static routes.

A. Directly connected networks

Directly connected networks are automatically placed in local RIBs. There are two methods to define directly connected route:

- An interface is configured with a valid IP address and mask. Such configuration is implicitly considered as a directly connected route that will be installed in routing table.
- A static route is configured without defining a next hop Ip address. It means only an outgoing interface is defined.

B. Static routes

Static routes govern packet switching on a local router. Implicitly, they have assigned low administrative value, which stands for their high priority in the forwarding process. This means that static routes will replace dynamic routes in the router's forwarding information base (FIB). For instance, using RIP dynamic routing protocol, router `R1` knows about the destination network `10.151.14.0/8` via interface `s0/1` with cost 8. However, static route for this destination suggests to use interface `s0/2`. The static route has priority over the dynamic route and, thus, router will send packets for `10.151.14.0/8` out the interface `s0/2`.

C. Representing Static Network RIB

Figure 2 depicts an example of static routing configuration and the resulting RIBs for networks N_1 and N_4. The computation starts with adjacency matrices that capture an effect of `ip route` configuration commands. Then, using a standard graph algorithm, e.g., Floyd-Warshall, the path matrices are computed. Because network N_1 is directly connected to router R_1 the column R_1 of the path matrix for N_1 defines all paths from any network destination to network N_1. From the example it can be concluded that

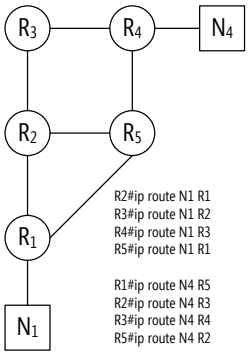
- it is possible to use standard graph approach to determine paths taken by packets routed under the static routing configuration, and
- for each network it seems to be necessary to perform independent comutation unless two or more networks share the same adjacency matrix.

Computing all pair shortests path using Floyd-Warshall algorithm has time complexity of $|V|^3$. As stated above we need to perform up to $|N|$ executions of this algorithm, which is the number of destinations to be analyzed, to determine the reachability of the network. Hence, the overall complexity is $|N| \cdot |V|^3$.

IV. Distance Vector Protocols

Routing processes running on network devices execute a distributed algorithm to collect all relevant routing data. These data are stored in a local database of routing protocol, which is private to the routing process. Every instance of routing protocol has its own private data structure that maintains this kind of information. The following is a content of a RIP database captured on a Cisco router:

```
Router#show ip rip database
11.0.0.0/8  directly connected, FastEthernet0/0
12.0.0.0/8
    [1] via 192.168.1.2, 00:00:23, Serial2/0
13.0.0.0/8
    [2] via 192.168.1.2, 00:00:23, Serial2/0
192.168.1.0/24    directly connected, Serial2/0
192.168.2.0/24
    [1] via 192.168.1.2, 00:00:23, Serial2/0
```



Static adjacency for N1

N1	R1	R2	R3	R4	R5
R1	0	x	x	x	x
R2	1	x	x	x	x
R3	x	1	x	x	x
R4	x	x	1	x	x
R5	1	x	x	x	x

Path matrix for N1

N1	R1	R2	R3	R4	R5
R1	x	x	x	x	x
R2	2,1:1	x	x	x	x
R3	3,2,1:2	3,2:1	x	x	x
R4	4,3,2,1:3	4,3,2:2	4,3:1	x	x
R5	5,1:1	x	x	x	x

Static adjacency for N4

N4	R1	R2	R3	R4	R5
R1	x	x	x	x	1
R2	x	x	1	x	x
R3	x	x	x	1	x
R4	x	x	x	0	x
R5	x	1	x	x	x

Path matrix for N4

N4	R1	R2	R3	R4	R5
R1	x	1,5,2:2	1,5,2,3:3	1,5,2,3,4:4	1,5:1
R2	x	x	2,3:1	2,3,4:2	x
R3	x	x	x	3,4:1	x
R4	x	x	x	x	x
R5	x	5,2:1	5,2,3:2	5,2,3,4:3	x

Figure 2. An example of static network configuration and network RIBs.

On the left side of the figure, there is a simple network topology consisting of five routers and static routing configuration snippets. On the right side there are adjacency matrices and path matrices for destinations N_1 and N_4. The path information contains also costs, which is nevertheless irrelevant in case of static configuration. The required result is obtained by taking column R_1 and R_4, respectively.

Using defined criteria, the router selects from local RIBs of all running routing protocols the information about available routes and installs them in the routing table. The content of a routing table determines paths, which the data would go through in the network.

Maltz et al. [13] developed a model for understanding routing contribution to network dynamics. In this section, we could employ their routing process graph to define an abstraction for network wide-routing information dissemination. However, we diverge from that approach as our goal is to define a method that would be able to compute efficiently the approximation of routing base information for the given network state without simulating (distributed) routing protocol algorithms.

For basic cases it is easy to determine routing information. Based on routing process graph one knows the flow restrictions of routing information and, by application of a standard graph traversal algorithm, it is possible to find the best paths with respect to cost models used by analyzed routing protocols. However, it is possible that routing information can be modified as it is spreading among routers. In this section, we deal with the case, in which the routing configuration includes access control lists applicable to routing information updates. In this case, to determine routing information bases it requires some additional steps.

A. Filtering Routing Updates

Route filtering is provided by regulating the route advertisments sent to neighboring routers and by filtering routes advertised by other routers before they are added to or updated in the local routing protocol database. Route filters have only an effect on distance vector protocols [14], e.g. Routing Information Protocol (RIP), Interior Gateway Routing Protocol (IGRP), and Enhabced IGRP (EIGRP).

It is possible to block routing updates sent through the interface or to control the processing and advertising of routes in routing updates. The first option stands for completely denying updates usually sent by the router to its neighbors through the connecting interface. The second option stands for applying filters that delete some routes from the routing update sent to the neighbor routers.

Depending on the device vendor, some form of access control lists is used to determine, which routes will be filtered. It is possible to process i) incoming routing updates to control, which routes are added to a local database, or ii) outgoing routing updates to control, which routes are sent to neighbor routers. Bellow you can se an example of two routing filters:

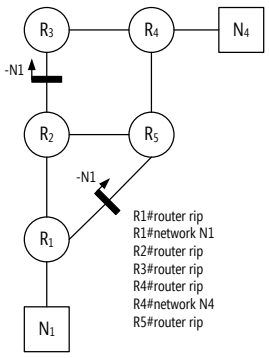
```
access-list 1 permit 1.0.0.0 0.255.255.255
access-list 2 permit 1.2.3.0 0.0.0.255
router rip
  distribute-list 2 in ehternet 0
  distrubute-list 1 out
```

The routing filter implemented by access control list 1 affects all outgoing routing updates and allows to send only information about destination `1.0.0.0/8`. The routing filter implemented by access control list 2 accepts from all updates received on etherent interface only information about destination `1.2.3.0/24`.

B. Computing the Effects of Filtering Routing Updates

First we consider the case without route filters. It is possible to easily compute a network RIB for the routing protocol instance by defining an adjacency table that exactly follows the neigborship relations in the network among the same routing protocol instances. Then, again by the using the standard graph algorihtms, the path matrix is computed. It would be possible to compute a single RIB that defines reachability and path information for all networks.

Then, if we consider the case, in which route filters could remove some destinations from routing updates, is is necessary to compute the path matrices for individual networks. If there is route filtering for network N on a link between router R_s and R_t the adjacency between these two nodes from the adjacency table must be removed. This



Routing adjacency for N1

N1	R1	R2	R3	R4	R5
R1	0	1	x	x	1
R2	1	x	1	x	1
R3	x	x	x	1	x
R4	x	x	1	x	1
R5	x	1	x	1	x

Path matrix for N1

N1	R1	R2	R3	R4	R5
R1	x	1,2:1	1,2,3:2	1,5,4:2	1,5:1
R2	2,1:1	x	2,3:1	2,3,4:2	2,5:1
R3	3,4,5,2,1:4	3,4,5,2:3	x	3,4:1	3,4,5:2
R4	4,5,2,1:3	4,5,2:2	4,3:1	x	4,5:1
R5	5,2,1:2	5,2:1	5,2,3:2	5,4:1	x

Routing adjacency for N4

N4	R1	R2	R3	R4	R5
R1	x	1	x	x	1
R2	1	x	1	x	1
R3	x	1	x	1	x
R4	x	x	1	x	1
R5	1	1	x	1	x

Path matrix for N4

N4	R1	R2	R3	R4	R5
R1	x	1,2:1	1,2,3:2	1,5,4:2	1,5:1
R2	2,1:1	x	2,3:1	2,3,4:2	2,5:1
R3	3,2,1:2	3,2:1	x	3,4:1	3,2,5:2
R4	4,5,1:2	4,3,2:2	4,3:1	x	4,5:1
R5	5,1:1	5,2:1	5,2,3:2	5,4:1	x

Figure 3. An example of route filtering and computation of network RIBs:
For simplicity there are only two route filters applied on links $\langle R_1, R_5\rangle$ and $\langle R_2, R_3\rangle$, respectively. These filters deny to send information on network N_1 to routers R_5 and R_3, which is captured in adjacency table for network N_1 by deleting adjacencies at $Adj_{N_1}[R_5, R_1]$ and $Adj_{N_1}[R_3, R_2]$. For comparison, Adj_{N_4} enjoys full adjacency as no filters for N_4 are configured in the network.

approach means to compute a path matrix for each network, which leads to $|V|^3 \cdot |N|$ time complexity, be the same as in the case of computing static network RIB. However, in real world scenario not all networks are filtered or a single filter affects multiple networks. Therefore, we can improve the performance by collecting all networks that are treated in the same way. We use term *slice* to denote such a collection of the equally treated networks.

A *slice* describes a collection of networks that are filtered by the same filters. We can compute a collection of slices by analyzing configuration of routing update filters. We use a set representation for a routing update filter, e.g. $f = \{n1, n2, n5, n7\}$ is a filter f that deletes information about networks n_1, n_2, n_5 and n_7 from an update. As decribed in the previous subsection, there are many kinds of filters applicable in various way. However, all these filters can be appropriately represented in a graph model by associating the filtering sets to edges.

Given F to be a set of filtering, we define the slicing S to be a set partitioning such that:

- $\forall f \in F, s \in S : s \cap f \neq \emptyset \implies s \subseteq f$,
- $\bigcup_{s \in S} s = N$
- $\forall s_1, s_2 \in S : s_1 \cap s_2 = \emptyset$.

The perfect slicing stands for such slicing that provides minimal $|S|$ with respect to F among a set of possible slicing sets. The problem is solved in the following two steps:

- for each network compute a set of filters in which the network occurs, and
- group networks according to their sets of filters; networks that have the same set of filters belong to the same slice.

Figure 3 depicts a demonstration of route filtering for two destination networks. For simplicity, we do not consider the slice-based approach. The algorithm starts by initiating cost matrices using information from configurations. Cost adjacency matrices for networks N_1 and N_4 describe the effect of routing update filters on a RIP databases. The basic idea is to classify edges between RIP vertices into two categories. If an edge has no associated filter that would prevent an information about a network to be sent from a source vertex to a destination vertex, then the edge has associated its cost; hence, this costs is stored in the cell of adjacency matrix. On the other hand, we use cost `x` to express that information is filtered on that edge. The path matrix is computed using the standard graph algorithm and we obtain N_1-RIB and N_4-RIB associated with RIP instance.

V. Redistribution

The situation when routing protocols advertise routes learned by some other means is called redistribution. While the most obvious and desirable use of Interior Gateway Protocols (IGP) is to employ a single scalable routing protocol for the entire domain, there are situations, which are quite common in practice: multiple IGP in a single domain. In these situations some form of redistribution is necessary. An introduction to redistributing routing protocols can be found in [15].

When redistributing, it is important to define a correct initial (seed) metric for each redistributing route as each protocol uses a different cost scheme. A configuration snippet bellow shows redistributing static routes and OSPF routes to RIP. A value after `metric` keyword specifies the seed metric used for redistributed routes in RIP.

```
router rip
redistribute static metric 1
redistribute ospf metric 1
```

A router uses an adminstrative distance to select the best route for each destination. This route is installed in the router's forwarding information base (routing table). Using

redistribution in a wrong way can lead to problems in forming routing loops, convergence problems or inefficient routing [16].

The redistribution mechanism is vendor dependend, but most platforms obey two additional rules when doing redistribution:

RR1: The route can only be redistributed if it is installed in router's FIB.

RR2: Even if a route is redistributed in the routing process with lower AD, this new route is not installed into router's FIB.

These two rules cause that redistribution is not transitive as pointed out by Le, Xie and Zhang in [16]. This observation makes our computation more complicated.

A. Computing Redistribution

To demonstrate the approach to redistribution we refer to the example shown in Figure 4. Redistribution is done within router's boundary. We define a redistribution matrix, which is similar to a matrix that denotes adjacency for distance vector routing protocols. This matrix expresses how the route redistribution is configured on a router, see Figure 4 b) for example.

Redistribution configured on router R_v, which redistributes routing information from RIB_s to RIB_t requires to insert paths at row v of matrix RIB_s to corresponding items in matrix RIB_t if these paths have lower costs and can be found in FIB. This is depicted by arrows between matrices 4 d) and 4 e) in the example.

Formally, the redistribution algorithm is defined as follows:

for all $r \in V$ **do**
  if $\mathsf{RIB}_s[v, r]$ is in FIB **then**
    $\mathsf{RIB}_t[v, r] = c_{min}(m_S(\mathsf{RIB}_s[v, r]), \mathsf{RIB}_t[v, r])$
  end if
end for

There c_{min} is selecting from two paths that path with the lower cost. Function m_S assigns a metric seed to the redistributing information.

The impact of redistribution rules RR1 and RR2 is best observable in the process of route selection. The route selection finds the most appropriate information from local RIBs and puts it in the router's FIB. Redistribution then must check whether the information that can be redistributed was selected for FIB to conform with rule RR1. To satisfy rule RR2, the route selection must not choose redistributed network into the FIB. The next section describes the route selection process in detail.

VI. Route Selection

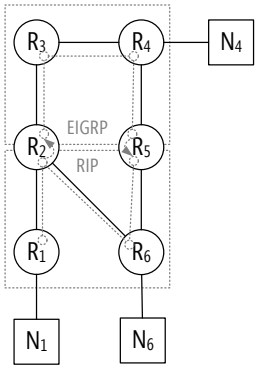
The purpose of the routing processes running at each router is to maintain routing protocol specific information. Every process manage its own (topological) database that allows to determine the best path to the destination as viewed by the routing protocol. The router needs to select a single (or a collection of alternate paths for load balancing) route to its FIB. This process is vendor dependent, but most often the routing information is prioritized by using administrative distance measure. This section provides a basic and straightforward algorithm that computes a content of the network FIB from a collection of network RIBs. For simplicity, we consider all routers follow the same route selection rules and no router contains redefined default administrative distance for any routing protocol nor any single route. This corresponds to the vast majority of configurations used in enterprise networks. The case when administrative distances are redefined is left for future work. The route selection algorithm is defined as follows:

for all $n \in N$ **do**
  for all $R \in \mathsf{RIB}$ **do**
    for all $r \in V$ **do**
      if $\exists p \in R[r, n]$ **then**
        $\mathsf{FIB}[r, n] = p$
        for all $s \in V : \langle q_0, \ldots, r, \ldots, q_n \rangle = \mathsf{FIB}[s, n]$ **do**
          $\mathsf{FIB}[s, n] = \langle q_0, \ldots, r \rangle + p.$
        end for
      end if
    end for
  end for
end for

The algorithm's time complexity is $|N| \cdot |R| \cdot |V^2|$, where N is a number of destination networks, R is a number of RIBs and V is a number of routers. Informally, the computation proceeds as follows:

1) Take the lowest priority network RIB and copy all information to a network FIB. This will initialize the network FIB.
2) Take the RIB with immediately higher priority and replace paths in FIB with existing paths in this RIB. This corresponds to the selection of route information with less administrative distance. If there is not path in this RIB then the path from a lower priority RIB remains in the FIB.
3) For each path in the RIB we must also check if this path can replace a suffix in an existing path in the FIB. This means, that if the FIB contains a path $\langle r_1, r_4, r_2, r_5, r_3 \rangle$ and the RIB contains a path $\langle r_4, r_7, r_8, r_9, r_3 \rangle$ we should replace the FIB's path with $\langle r_1, r_2, r_4, r_7, r_8, r_9, r_3 \rangle$. This replacement corresponds to installation of route with lower AD in a router on the existing path.
4) Repeat from step 2 until we process all RIBs.

Information stored in the network FIB can be used to determine paths to all destinations. In our example presented in Figure 3 we compute the network $N^\star$-FIB for all networks



(a) Network topology graph

Redistribution matrix

R2	Static	RIP	EIGRP
Static	x	x	x
RIP	x	x	1
EIGRP	x	x	x

Redistribution matrix

R5	Static	RIP	EIGRP
Static	x	x	x
RIP	x	x	x
EIGRP	x	1	x

(b) Redistribution

EIGRP

N*	R1	R2	R3	R4	R5	R6
R1	x	x	x	x	x	x
R2	x	x	1	x	x	x
R3	x	1	x	1	x	x
R4	x	x	1	x	1	x
R5	x	x	x	1	x	x
R6	x	x	x	x	x	x

RIP

N*	R1	R2	R3	R4	R5	R6
R1	x	1	x	x	x	x
R2	1	x	x	x	x	1
R3	x	x	x	x	x	x
R4	x	x	x	x	x	x
R5	x	x	x	x	x	1
R6	x	1	x	x	1	x

(c) Routing adjacency

EIGRP RIB

N*	R1	R2	R3	R4	R5	R6
R1	x	x	x	x	x	x
R2	x	x	2,3:1	2,3,4:2	2,3,4,5:3	x
R3	x	3,2:1	x	3,4:1	3,4,5:2	x
R4	x	4,3,2:2	4,3:1	x	4,5:1	x
R5	x	5,4,3,2:3	5,4,3:2	5,4:1	x	x
R6	x	x	x	x	x	x

RIP RIB

N*	R1	R2	R3	R4	R5	R6
R1	x	1,2:1	x	x	1,2,6,5:3	1,2,6:2
R2	2,1:1	x	x	x	2,6,5:2	2,6:1
R3	x	x	x	x	x	x
R4	x	x	x	x	x	x
R5	5,6,2,1:3	5,6,2:2	x	x	x	5,6:1
R6	6,2,1:2	6,2:1	x	x	6,5:1	x

FIB

N*	R1	R2	R3	R4	R5	R6
R1	x	1,2:1	x	x	1,2,6,5:3	1,2,6:2
R2	2,1:1	x	2,3:1	2,3,4:2	2,3,4,5:3	2,6:1
R3	x	3,2:1	x	3,4:1	3,4,5:2	x
R4	x	4,3,4:2	4,3:1	x	4,5:1	x
R5	5,6,2,1:3	5,4,3,2:3	5,4,3:2	5,4:1	x	5,6:1
R6	6,2,1:2	6,2:1	x	x	6,5:1	x

(d) Network RIBs and the network FIB

EIGRP with RIP redistributed routes

N*	R1	R2	R3	R4	R5	R6
R1	x	x	x	x	x	x
R2	2,1:1	x	2,3:1	2,3,4:2	2,3,4,5:3	2,6:1
R3	x	3,2:1	x	3,4:1	3,4,5:2	x
R4	x	4,3,2:2	4,3:1	x	4,5:1	x
R5	x	5,4,3,2:3	5,4,3:2	5,4:1	x	x
R6	x	x	x	x	x	x

RIP with EIGRP redistributed routes

N*	R1	R2	R3	R4	R5	R6
R1	x	1,2:1	x	x	1,2,6,5:3	1,2,6:2
R2	2,1:1	x	x	x	2,6,5:2	2,6:1
R3	x	x	x	x	x	x
R4	x	x	x	x	x	x
R5	5,6,2,1:3	5,4,3,2:1	5,4,3:1	5,4:1	x	5,6:1
R6	6,2,1:2	6,2:1	x	x	6,5:1	x

(e) RIBs with redistributed routes

EIGRP RIB with redistributed RIP routes

N*	R1	R2	R3	R4	R5	R6
R1	x	x	x	x	x	x
R2	~~2,1:1~~	x	2,3:1	2,3,4:2	2,3,4,5:3	~~2,6:1~~
R3	3,2,1:2	3,2:1	x	3,4:1	3,4,5:2	3,2,6:2
R4	4,3,2,1:3	4,3,2:2	4,3:1	x	4,5:1	4,3,2,6:3
R5	5,4,3,2,1:4	5,4,3,2:3	5,4,3:2	5,4:1	x	5,4,3,2,6:4
R6	x	x	x	x	x	x

RIP RIB with redistributed EIGRP routes

N*	R1	R2	R3	R4	R5	R6
R1	x	1,2:1	1,2,6,5,4,3:4	1,2,6,5,4:4	1,2,6,5:3	1,2,6:2
R2	2,1:1	x	2,6,5,4,3:3	2,6,5,4:3	2,6,5:2	2,6:1
R3	x	x	x	x	x	x
R4	x	x	x	x	x	x
R5	5,4,3,2,1:2	5,4,3,2:1	5,4,3:1	5,4:1	x	5,6:1
R6	6,2,1:2	6,2:1	6,5,4,3:2	6,5,4:2	6,5:1	x

FIB

N*	R1	R2	R3	R4	R5	R6
R1	x	1,2:1	1,2,6,5,4,3:4	1,2,6,5,4:4	1,2,6,5:3	1,2,6:2
R2	2,1:1	x	2,3:1	2,3,4:2	2,3,4,5:3	2,6:1
R3	3,2,1:2	3,2:1	x	3,4:1	3,4,5:2	3,2,6:2
R4	4,3,2,1:3	4,3,2:2	4,3:1	x	4,5:1	4,3,2,6:3
R5	5,4,3,2,1:4	5,4,3,2:3	5,4,3:2	5,4:1	x	5,4,3,2,6:4
R6	6,2,1:2	6,2:1	6,5,4,3:2	6,5,4:2	6,5:1	x

(f) Converged network RIBs and the network FIB

Figure 4. An example of route redistribution and selection:
Initially, network RIBs for two routing instances are computed as showed in (d). The computation gets as an input the data from redistribution matrices (b), which define redistribution flows within routers and routing adjacency matrices (c), which define routing information flow between routers. The redistribution requires to compute network FIB such that the redistribution rules can be checked. Then the redistribution copies appropriate information from RIP to EIGRP and from EIGRP to RIP, which is shown in (e). After that the RIBs are updated and FIB is determined, see (f). In (d) and (f), the underlined items stand for RIB's paths selected for FIB. In (e) the underlined items stand for paths redistributed from the source routing protocol.

as we do not consider static routing nor route filtering. If we analyze network configuration that combines static routing, dynamic routinng with route filtering and redistribution, we would need to compute more network FIBs depending on the number of networks. Alternatively, it is possible to reduce a number of computations by creating slices as explained in section IV.

VII. Conclusions

In this paper we define a method for computing the network-wide view of the forward information base (FIB), which allows us to predict all delivering paths in the network. This work is complementary to the work on packet filter analysis as carried out by, e.g., Guttman [3], Liu [7], Bera, Dasgupta and Ghosh [10]. The work presents

an alternative method to the approach presented by Xie, Zhan, Maltz and Zhang in [2] and, in particular, by Maltz, Xie, Zhan and Greenberg [13]. Contrary to their work, we compute the global view using standard graph algorithms without the need to mimic behaviors of routing protocols. On the other hand, it can become difficult to represent a route modification, e.g., tagging used to prevent routing loops. While we have determined the theoretical complexity of the method, the future work is required to assess its practical contribution on real world examples.

ACKNOWLEDGMENT

This work was partially supported by the BUT FIT grant FIT-10-S-2 and the research plan MSM0021630528 and by the Grant Agency of the Czech Republic through the grant no. GACR 102/08/1429: Safety and Security of Networked Embedded System Applications. Also, the first co-author was supported by the grant no. FR-TI1/037 of Ministry of Industry and Trade: Automatic Attack Processing.

REFERENCES

[1] N. Feamster, "Practical verification techniques for wide-area routing," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 87–92, 2004.

[2] G. G. Xie, J. Zhan, D. A. Maltz, H. Zhang, A. Greenberg, G. Hjalmtysson, and J. Rexford, "On static reachability analysis of ip networks," in *in Proc. IEEE INFOCOM*, 2005.

[3] J. D. Guttman, "Filtering postures: Local enforcement for global policies," in *In Proceedings, 1997 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1997, pp. 120–129.

[4] L. Yuan and H. Chen, "Fireman: a toolkit for firewall modeling and analysis," in *In Proceedings of IEEE Symposium on Security and Privacy*, 2006, pp. 199–213.

[5] A. Jeffrey and T. Samak, "Model checking firewall policy configurations," *Policies for Distributed Systems and Networks, IEEE International Workshop on*, vol. 0, pp. 60–67, 2009.

[6] S. Pozo, R. Ceballos, and R. Gasca, "Fast algorithms for consistency-based diagnosis of firewalls rule sets," in *Proceedings of the 3rd International Conference on Availability, Reliability and Security (ARES)*, 2008.

[7] A. X. Liu, "Formal verification of firewall policies," in *Proceedings of the 2008 IEEE International Conference on Communications (ICC)*, Beijing,China, May 2008.

[8] M. Gouda, A. X. Liu, and M. Jafry, "Verification of distributed firewalls," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, New Oreleans, Louisiana, Novembrt 2008.

[9] A. X. Liu and M. G. Gouda, "Firewall policy queries," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, to appear.

[10] P. Bera, S. Ghosh, and P. Dasgupta, "Formal verification of security policy implementations in enterprise networks," in *ICISS '09: Proceedings of the 5th International Conference on Information Systems Security*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 117–131.

[11] ——, "Formal analysis of security policy implementations in enterprise networks," *International journal of Computer Networks and Communications*, vol. 1, no. 2, pp. 56–73, 2009.

[12] P. Matousek, J. Rab, O. Rysavy, and M. Sveda, "A formal model for network-wide security analysis," in *Proceeding of the 15th IEEE International Symposium and Workshop on the Engineering of Computer-based Systems*. University of Ulster, 2008, pp. 171–181.

[13] D. A. Maltz, G. Xie, J. Zhan, H. Zhang, and A. Greenberg, "Routing design in operational networks: A look from the inside," in *In Proc. ACM SIGCOMM*, 2004.

[14] "Filtering routing updates on distance vector ip routing protocols," Cisco Systems, Document ID:9105, September 2006.

[15] "Redistributing routing protocols," Cisco Systems, Document ID:8606, September 2006.

[16] F. Le, G. Xie, and H. Zhang, "Understanding route redistribution," in *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, 2007, pp. 81 –92.