Routing Design in Operational Networks: A Look from the Inside *

David A. Maltz, Geoffrey Xie, Jibin Zhan, Hui Zhang
{dmaltz,geoffxie,jibin,hzhang}@cs.cmu.edu
Carnegie Mellon University

Gísli Hjálmtýsson†, Albert Greenberg
{gisli,albert}@research.att.com
AT&T Labs–Research

Abstract

In any IP network, routing protocols provide the intelligence that takes a collection of physical links and transforms them into a network that enables packets to travel from one host to another. Though routing design is arguably the single most important design task for large IP networks, there has been very little systematic investigation into how routing protocols are actually used in production networks to implement the goals of network architects. We have developed a methodology for reverse engineering a coherent global view of a network's routing design from the static analysis of dumps of the local configuration state of each router. Starting with a set of 8,035 configuration files, we have applied this method to 31 production networks. In this paper we present a detailed examination of how routing protocols are used in operational networks. In particular, the results show the conventional model of "interior" and "exterior" gateway protocols is insufficient to describe the diverse set of mechanisms used by architects. We provide examples of the more unusual designs and examine their trade-offs. We discuss the strengths and weaknesses of our methodology, and argue that it opens paths towards new understandings of network behavior and design.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]

General Terms

Design, Management, Measurement

*This research was sponsored by the NSF under ITR Awards ANI-0085920 and ANI-0331653. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of AT&T, NSF, or the U.S. government.

†Also at Reykjavík University

SIGCOMM'04, Aug. 30–Sept. 3, 2004, Portland, Oregon, USA.

Keywords

Routing design, static configuration analysis, reverse engineering, network modeling

1. Introduction

By constructing the collective distributed routing state, routing protocols create the "network-wide intelligence" that transforms a collection of individual links and routers into an IP network. A network's *routing design* is embodied in the configuration of these protocols. While originally targeted at establishing basic reachability, in practice routing designs are used to attempt to deal with a large and complex set of objectives and constraints: (i) providing resiliency and predictable behavior under a wide set of internal or external fault or overload conditions; (ii) maintaining stable and efficient internal operations; (iii) maintaining contractual or business relationships between different administrative domains; (iv) coping with complex interactions between a wide set of protocols, which run concurrently, overlap in functionality, and collectively determine the forwarding tables within the routers.

Creating a routing design is in practice a policy driven design task of specifying packet filters, link weights, routing policies, and so forth. Understanding a routing design is complicated by the enormous range of options the routing designer may choose from to realize given objectives and constraints – a diverse range of routing designs may all satisfy a given set of constraints. More importantly, intricate details of the design choices have significant impact on fundamental aspects of overall network performance and operations, including complexity, cost, and survivability. Routing design is both inherently hard and the single most important network design task.

It is natural to think of numerous possibilities to improve the situation: e.g., construction of simpler, more robust and more efficient routing designs with available protocols; construction of better models for reasoning about the range of emergent routing states that may result from the design in the operational network; construction of better configuration languages and better protocols that more cleanly separate policy intent from implementation, so that policies can be better composed and reasoned about. To succeed, we need first to get some level of understanding of what routing designs look like in operational networks, and what the routing designers are attempting to achieve. In practice this must be an exercise in reverse engineering, in part because documentation lags the network as network technologies and

business conditions change rapidly and overloaded operations staff only need the specific configuration rules rather than the global configuration intent.

Two reverse engineering methodologies might be attempted. One, which we term "black-box" involves using a variety of data available without administrative privilege, including trace routes, pings, BGP table dumps, and DNS lookups. An excellent set of "black-box" research results have come from the RocketFuel [25], Skitter [2], and Mercator [8] projects. These projects have provided some understanding of the topology and POP structure of major backbone networks, providing an estimated snapshot of the routers and their IP connectivity within a given administrative domain. Another approach, which we term "white-box" involves using data available to those with network administrative privileges. Insights and results from white-box approaches can go substantially beyond IP topology snapshots. Topology snapshots emerge as the results of the interactions of routing configuration and routing protocols. White-box approaches shed direct light on the routing design that governs the protocols that produce the snapshots, and provide fundamental data needed to reason about why a particular topology emerges. In contrast to black-box approaches, there is little understanding of the power and limitations of white-box approaches.

With this paper we begin the process of structuring and analyzing IP routing designs. Our approach is pragmatic. Rather than theorizing about goals and metrics, we have chosen to begin our research by investigating routing configurations of existing production networks. By far the best source of structure and design pattern information available for operational IP networks are the running configuration files associated with the routers. A router configuration file provides a dump of the complete set of configuration commands currently executing on the router. Roughly, a router configuration file corresponds to a program, and the set of router configuration files in a network corresponds to a distributed program. Just as program analysis has had a rich history (e.g., [16]) and a profound impact on computing technologies (e.g., modern compilers, RISC architectures), we believe routing design analysis may illuminate the way forward for better configuration languages and simpler, more robust network architectures.

In this paper, we propose a scalable white-box approach for reverse engineering of routing designs, from data easily and routinely archived today in virtually all operational networks. In collaboration with a major network service provider we retrieved and anonymized the configuration files of 23,417 production routers. With the aid of a trusted intermediary we selected for detailed analysis 8,035 configuration files constituting 31 production networks ranging in size from medium to large, representing regional enterprises, global enterprise networks, and segments of provider's backbone networks. To our knowledge such an undertaking has not been done before, and certainly not at a comparable scale. This is a significant contribution of this paper. An additional contribution is our methodology of working with anonymous data. Configuration files commonly contain sensitive and proprietary information. Working with anonymous data was key in getting access to the set of configuration files and makes our methods viable to the larger networking community. Using this methodology we perform a detailed examination of how routing protocols are used in operational networks and derive a number of interesting observations that could not have been made using any other existing approach.

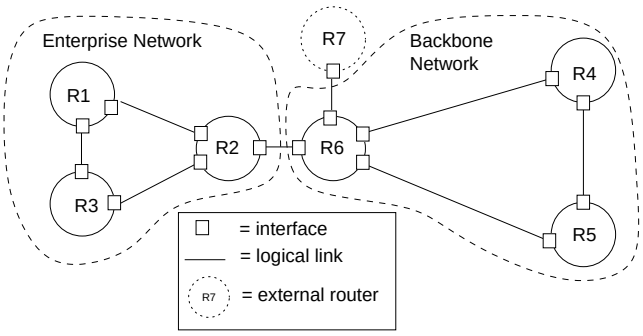

Figure 1: An example topology showing routers, interfaces, and links. R1-R3 represent a small enterprise network customer connected to R4-R6, which represent part of a transit network that also serves R7.

2. Background

This section explains the types of configurations required to implement a routing design. To make the explanation more concrete, Figure 1 shows a router-level view of an example topology. Routers R1 to R7, depicted as disks, are connected with physical links (shown as solid lines in the figure). Links terminate at interfaces, shown as small squares. In this example, routers R1-R3 belong to a small enterprise network that obtains connectivity to the Internet through a transit backbone network, of which routers R4-R6 are a part. Router R7 belongs to another customer of the backbone network. Figure 2 shows part of the routing configuration file from router R2 in Figure 1. This "configlet" is in the Cisco IOS language. While the syntax of other router configuration languages differ, the granularity and type of information they contain are very similar. We use this example and these figures throughout this section. As described in Section 4, user-specific information is anonymized for privacy reasons.

2.1 Link-level Topology

Each router has one or more interfaces; each interface has one or more IP addresses and *subnets* that identify the set of other IP addresses directly reachable from that interface. Lines 1-11 of Figure 2 show interface definitions for three interfaces of R2, an Ethernet, Serial, and High Speed Serial (Hssi) interface, having IP addresses and subnets 66.251.71.144/25, 66.253.32.85/30, and 66.253.160.67/30, respectively.

From the configuration files, we infer the logical IP links between routers by matching interfaces with the same subnet.[1] When an interface fails to match with any other interface in the network's configuration files, we can usually

[1]Interfaces can also be *unnumbered*, meaning that no IP address is assigned to them. These interfaces cannot easily be matched into links without additional information, but they are quite rare in the networks we have evaluated so far: we found only 528 unnumbered interfaces out of 96,487 total interfaces.

```
1   interface Ethernet0
2    ip address 66.251.75.144 255.255.255.128
3    ip access-group 143 in
4   !
5   interface Serial1/0.5 point-to-point
6    ip address 66.253.32.85 255.255.255.252
7    ip access-group 143 in
8    frame-relay interface-dlci 28
9   !
10  interface Hssi2/0 point-to-point
11   ip address 66.253.160.67 255.255.255.252
12  !
13  router ospf 64
14   redistribute connected metric-type 1 subnets
15   redistribute bgp 64780 metric 1 subnets
16   network 66.251.75.128 0.0.0.127 area 0
17  !
18  router ospf 128
19   redistribute connected metric-type 1 subnets
20   network 66.253.32.84 0.0.0.3 area 11
21   distribute-list 44 in Serial1/0.5
22   distribute-list 45 out
23  !
24  router bgp  64780
25   redistribute ospf 64 match route-map 8aTzlvBrbaW
26   neighbor 66.253.160.68 remote-as  12762
27   neighbor 66.253.160.68 distribute-list 4 in
28   neighbor 66.253.160.68 distribute-list 3 out
29  !
30  access-list 143 deny 134.161.0.0 0.0.255.255
31  access-list 143 permit any
32  route-map 8aTzlvBrbaW deny 10
33   match ip address 4
34  route-map 8aTzlvBrbaW permit 20
35   match ip address 7
36  ip route 10.235.240.71 255.255.0.0 10.234.12.7
```

Figure 2: Part of the router configuration file from R2 in Figure 1 showing three interface definitions, two different instances of OSPF, one instance of BGP, and assorted policies. User-specific information, such as a IP addresses and route-map names, have been anonymized for privacy (e.g., the route-map name "8aTzlvBrbaW" is a random string that replaces the actual name).

declare the interface to be *external facing* and anything connected to it must be *external* to the network. In Figure 1, R6 is external to the enterprise network, and R7 is external to both enterprise and backbone networks.

2.2 Routing Topology

Routing protocols are typically classified as either Interior Gateway Protocols (IGPs) used to exchange information inside a network (e.g., OSPF [20], IS-IS [4], RIP [13], and EIGRP [26]) or an Exterior Gateway Protocol (EGP) used to exchange information between networks (e.g., BGP [22]). Both IGPs and EGPs share the common goal of exchanging routing information between routers, but differ in the features and performance they provide. Each router can use multiple protocols simultaneously; moreover multiple instances of the same protocol may exist on a single router. To maintain boundaries on how routing information is shared, each routing protocol runs as a separate process on the router and is identified by a process-id. In Figure 2, lines 13-16, 18-22, and 24-28 define routing process 64 speaking OSPF, routing process 128 speaking OSPF, and routing process 64780 speaking BGP, respectively. By default, no information is exchanged between these routing processes.

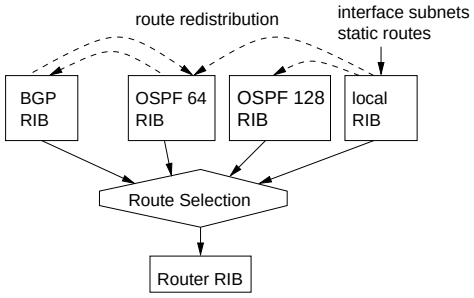

Figure 3: The relation between routing process RIBs, route creation, route redistribution, and the router RIB that stores routes used to forward packets.

Each routing process can be associated with one or more interfaces on the router. There are many ways to create this association, but the most common one is via a `network` command (e.g. line 16 in Figure 2) that covers the address assigned to the interface (line 2).

For two routing processes on different routers to *directly* exchange routing information, the processes must be *adjacent*. The definition of adjacent depends on the type of the routing process. Two BGP processes are adjacent if the processes are explicitly configured to speak to each other and it is possible to open a TCP connection between the two routers. For OSPF, IS-IS, RIP, or EIGRP processes to be adjacent, the processes must be of the same type; there must be a link between the routers on which the processes run; and each process must be configured to cover the interface at its end of the link. If R4 and R5 in Figure 1 are running OSPF processes and both interfaces on those routers are associated with the process, then the OSPF processes on R4 and R5 would be adjacent.

2.3 Route Calculation and Selection

We model a *route* as an IP subnet address (e.g., 10.0.0.0/8) plus some additional attributes, such as weights or an AS path, that the router may use to calculate a next-hop to reach that subnet. There are several ways a router can learn a route. Routes to all the directly connected subnets are always available to the router, or routes can be manually configured (e.g., with static routes). An example of a static route is line 36 in Figure 2.

Through routing protocols, routes can be learned dynamically. While different protocols exchange different types of routing information to convey routes between adjacent processes, e.g., OSPF and IS-IS use link-state advertisements and BGP uses path-vector records, the end result is the processes learning routes.

For the purpose of reasoning about routing designs, the details of a large class of routers can be abstracted to the model depicted in Figure 3, where each routing process maintains its own Routing Information Base, RIB, where its associated routing state is stored. The routes a router uses for forwarding packets are centrally stored in the *router RIB*. Route selection logic is used to select which routes from the routing process RIBs should be entered into the router RIB. Prior work [7, 9, 10, 21] has studied route selection in BGP.

However, BGP route selection determines only which routes are present in the RIB of the BGP process. A second route selection process determines which of those routes are entered into the router RIB.

2.4 Redistribution, Routing Policies and Packet Filtering

Routing protocols exchange routing information, hence routes, between routers. Inside a single router, a mechanism called *route redistribution* is used to transfer routes between routing processes, as illustrated by the dashed arrows in Figure 3. [2]

To model the handling of routes for the subnets that are directly connected to the router and routes that are statically configured, we introduce a *local RIB* that holds these routes. This makes the handling of static routes parallel to that of dynamically learned routes, as route redistribution can then be used to redistribute routes from the local RIB into the other routing processes on the router. Lines 14 and 19 are examples of this type of redistribution.

Routing policies are the mechanisms that control the exchange of routes between routers and between routing processes on the same router. Modern routers support a rich language to specify routing policies, and the complexity of routing design is largely incorporated into these policies.

In our example above, R2 uses "distribute-list 4" to control the routes learned from R6, and "distribute-list 3" for routes announced out. It uses route-map "8aTzlvBrbaW" to control which routes can be redistributed from the "ospf 64" routing process into the BGP process.

Redistribution and routing policy operate in the control plane of the network and determine the path that packets will take from their sources to their destinations. There is another kind of policy control in the network which works directly on the data plane. That is *packet filtering*. Packet filtering enables a router to classify the incoming or outgoing packet stream based on the properties associated with individual packets or packet streams. Matching packets are either forwarded (allowed) or dropped (denied).

Unlike routing information, packet filters are interface specific, statically configured, are not shared across routers, and can only be changed manually. In Figure 2, lines 30 to 31 define a packet filter, which is assigned to the Serial1/0.5 interface in line 7.

3. A Model for Understanding Routing Design

While router configuration files are designed to be editable directly by human operators, it is extremely tedious to reverse engineer the routing design of a network by manually extracting information from the configuration files. Many production networks are large in terms of both the number of routers they contain and the size of each router's configuration files. Figure 4 shows the configuration file size distribution of one network in our data set, which has a total of 881 routers. The configuration files for these routers contain an average of 270 lines of configuration commands each. With a total of 237,870 commands used to configure

[2]JunOS and Gated use import and export commands, which always go through the router RIB, but this can be modeled in our framework.

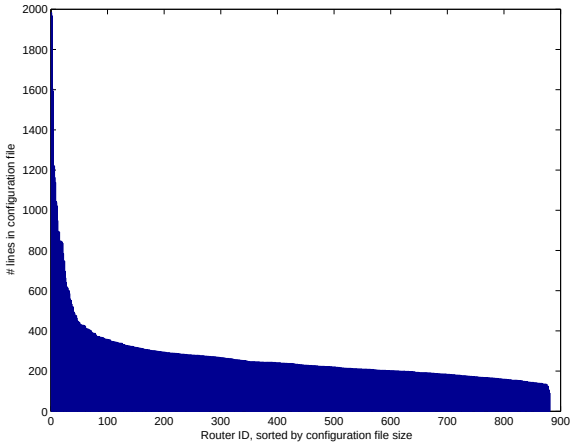

Figure 4: Size distribution of the configuration files for net5

the network, examining even 10% of them by hand would be a major challenge.

More importantly, the number of details in each configuration file and the distributed nature of router configuration makes manual reverse engineering unreliable. Correctly extracting the routing design requires creating the proper relationships between literally thousands of details, some located inside a single file and some distributed across many different files. It is like trying to capture by hand the global behavior of dozens of distributed yet interacting programs written in assembly language. What is needed is a framework for systematically reverse engineering, representing, and analyzing routing designs that enables the scientific study of the art of routing design.

In this section we describe four abstractions we have developed that can be automatically reverse engineered from a network's router configuration files: routing process graphs, routing instances, route pathway graphs, and address space structure. With these abstractions, we have a succinct means to capture the routing design of network and reduce the need for researchers and operators to work with routers only at the level of the configuration files. It opens the door to discussion of the performance and operation of complete networks, rather than individual protocols.

3.1 Routing Process Graphs

Our first step for extracting the routing design of a network is to build the *routing process graph* that models how routing information flows through the network. The vertices in this graph are the RIBs that store the routing information learned by each routing process. Since there is one RIB for each routing process on a router, the vertex list can be easily extracted from the configuration files. An edge between two RIBs is added to the graph whenever routes from one RIB might be transfered to the other RIB. These edges are discovered by parsing the configuration files for all commands that create adjacencies between routing processes or that import, export, or redistribute routes between processes. Policies that govern the exchange of routes can be modeled as annotations on the edges of the graph. Figure 5 shows the routing process graph for the example of

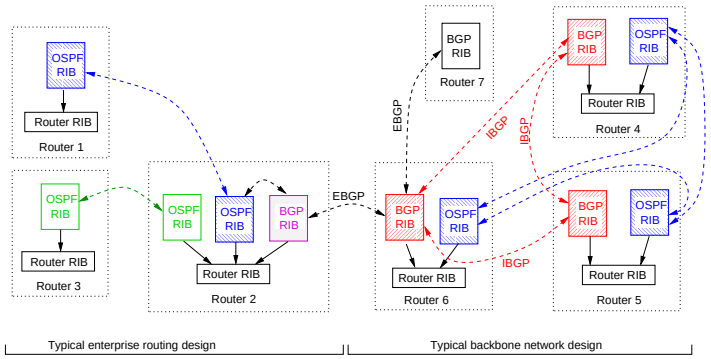

Figure 5: Routing process graph for the example networks in Figure 1.

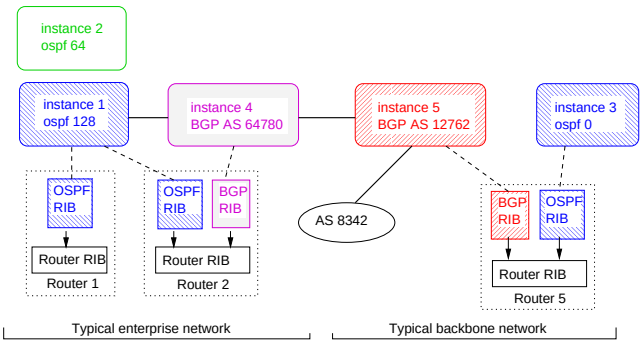

Figure 6: A depiction of the routing designs for the networks in Figures 1 and 5.

Figure 1, in which a small enterprise network is connected to a transit backbone network.

There are two main advantages of the routing process graph. First, it immediately becomes easier to frame questions that were previously unanswerable via static analysis, such as how many routes will a routing process have to handle or what destinations will be reachable from a particular router under any given failure scenario [27].

Second, the routing process graph exposes the detailed structure of the routing design so that alternative designs can be compared. For example, the enterprise network and backbone network in Figure 5 both contain the same number of routers, but have very different routing designs. In the backbone network (right half of the figure), routes to external subnets are learned from external peers (R7 and R2) via an External-BGP session (EBGP) and shared with the other routers in the network via Internal-BGP sessions (IBGP). The EBGP speaker also announces to the outside world the routes reachable via this AS. An IGP process (e.g., OSPF) is run on each router in the network and used to compute routes to all subnets internal to the network (aka infrastructure routes). This design is typical of large ISP transit networks. The hallmark of this design is that external routes are never redistributed into OSPF: the only place internal and external routes come together is in the router RIB of each individual router. Redistribution policies are not shown in the figure for clarity.

In the enterprise network (left half of the figure), routes to external subnets are again learned from an external peer (R6) via an External-BGP session (EBGP), but here they are redistributed into OSPF on the border router (R2). The OSPF processes then exchange routes to both internal and external destinations. This design is typical of small enterprise networks, where it is chosen because BGP processes only need to be configured on the border routers, which are few in number. This also minimizes the size of the IBGP mesh that must be constructed inside the network. The border routers use BGP's extensive routing policy features to craft a small number of key routes that summarize the external routes they have learned and inject these summaries into the IGP.

3.2 Routing Instance Graphs

Directly showing the relationship between the routing processes on different routers, as is done in a routing process graph (Figure 5), is tremendously valuable in understanding the interaction between those processes. However, as we have discovered in our analysis of the production networks, the value of this type of model drops rapidly as the number of routers and the complexity of the routing design increases. To further abstract away details that obscure the structure of the routing design and enable the analysis and understanding of larger networks, we introduce the concept of a *routing instance* that can represent a large number of routing processes.

A routing instance models the set of routing processes that share routing information. We compute the routing instances for a network from its configuration files by grouping together all the routing processes running the same protocol that are adjacent to each other. We first select from the network a routing process that has not yet been assigned an routing instance and assign to it a new unique instance number. We then locate all the adjacencies of that process, and compute the transitive closure to find the set of routers and routing processes belonging to the new routing instance. The closure operation flood fills through the routing process graph, stopping when it reaches an edge between routing processes of different types or an EBGP adjacency between BGP speakers with different AS numbers. The process of selecting an unassigned routing process is then repeated until all routing processes have been assigned to a routing instance.

As described in Section 2.2, each routing process has a process ID assigned to it by the configuration file. However, the meaning of these process IDs is entirely separate from the routing instance defined above. In many production networks we examined the process ID gives no indication of how the routing processes are connected. It is very common for routing processes with the same process ID to be found in two different routing instances, or for processes with different IDs to be found in the same routing instance. In general, the only requirement enforced on routing process IDs is that they be unique on each router — they have no network-wide semantics.

Figure 6 shows the result of applying our routing instance model to the example networks in Figure 5. The individual routers and routing processes in the networks have been removed and replaced with the routing instances that the routing processes are part of. The heavy lines between instances indicate where route exchange occurs between different protocols or ASs. In order to help a human reader

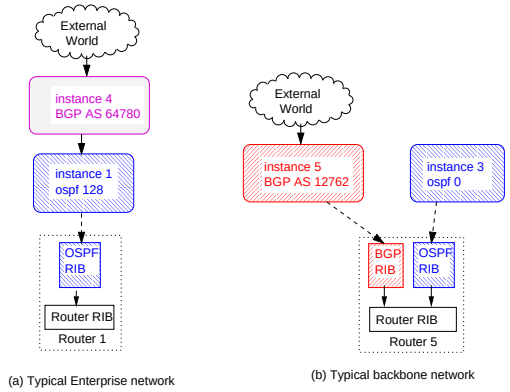

Figure 7: The pathways via which routing information is learned by Router 1 and Router 5. The pathway in (a) is typical of enterprise networks and the pathway in (b) is typical of backbone networks.

understand how a particular router fits into the routing design, that router can be added back into the model as shown by routers R1, R2, and R5 in Figure 6. Dashed lines are used to indicate the routing instance to which each routing process belongs.

The value of the routing instance model is its ability to scale to large numbers of routers without losing the ability to capture complex policies and interactions between multiple routing protocols. While in this example the routing process graph and routing instance graph have similar complexity, in later sections we will show examples from real networks where a single routing instance is able to represent hundreds of routing processes — radically reducing the complexity of the graph and rendering the structure of the network easily understandable.

3.3 Route Pathway Graphs

Using the routing instance model, we can construct for any router a *route pathway graph* showing where the routes used by that router come from. Starting at the Router RIB for the router in question, we perform a breath-first-search through the routing instance model, recording the instances through which the search passes. As shown in Figure 7, Router 1 learns all its routes from routing instance 1, which learns all its routes from instance 4, which learns routes from the outside world via an external peer. Router 5 learns routes from instances 5 and 3, and all routes to the external world must come via instance 5. Routing policies are typically applied at each edge in the pathway graph. Route pathways are useful for characterizing the routing design of a network. They can also be used to locate all the routing policies that affect the routes seen by any particular router, and pinpoint where the policies are applied.

3.4 Extracting the Address Space Structure

Network designers often have a structured plan for assigning addresses inside the network. Since the routes exchanged by the routing protocols are written in terms of subnets that represent blocks of IP addresses, understanding the structure used to divide the address space into blocks is very helpful in analyzing the routing design. Unfortunately, the configurations may never explicitly list the address blocks themselves, and without additional processing the subnets found inside the configurations are too small and fragmented to reveal the overall structure of the address space usage.

We recover the structure of the address space usage by finding a set of network numbers and netmasks that best covers the subnets that are mentioned in the configuration files. This discovery process starts with a list of all subnets mentioned in the network. The process then repeatedly joins together any two subnets whose network numbers differ in no more than the least two bits (basically, expanding the IP subnets so long as at least half the addresses in the enlarged subnet are "used" by the network) until no more joins are possible. The result is a hierarchical tree of address space blocks.

In extracting a network's routing design we make use of the address space structure in two ways. First, we can associate with each routing instance the set of address blocks that are connected to the instance. This reduces the number of individual subnets present in the extracted routing design and makes the design easier to understand and reason about. Second, it helps us to determine if the network being analyzed contains routers whose configuration state was left out of the data set. When a router R is missing from the data set, the routers in the data set will have their interfaces that should be adjacent to R erroneously marked as external-facing, since the matching interfaces on R cannot be found (see Section 2.1). Analysis of the address space structure can then help identify the missing router. Many networks assign their external-facing interfaces from a different block of addresses than the block used to assign internal-facing interfaces. If an interface is marked "external-facing" but has an address from the middle of an block used by many internal-facing interfaces, then it is very likely that the data set is missing a router's configuration file, and that "external-interface" is actually connected to the missing router.

4. Obtaining Network Configurations

The 31 networks whose configuration files we analyzed were obtained from a large telecommunications company that manages enterprise networks as part of its service portfolio. As is typical in real networks, the routing designs we see include those designed by the ISP's engineers, those designed by the ISP's customers, and hybrid mixes of the two.

The value of router configuration files to networking researchers is that they contain intricate details about the structure and operation of the network they describe. For this same reason, they are carefully guarded as proprietary secrets of the companies that own and manage the network they describe. This secrecy has hampered researchers from gaining access to network configurations, and has driven the development of black-box methods of network analysis as the only alternative. We have overcome this challenge by developing a means for anonymizing router configuration files that preserves enough structure in the files to enable our analysis, but prevents readers of the files from determining even the identity of the network being examined.

To conduct our study, we combined configuration file anonymization with a single-blind methodology. Three members of our group had knowledge of the identity of the networks being examined and contact information for the designers of

those networks, but this information was kept from the rest of the group. All analysis described in this paper was performed on the anonymized configuration files without knowledge of the network identity. Our results were then verified with the actual designers via the group members with the contact information for those networks.

Without both anonymization and the single-blind methodology, access to these configuration files would have been inconceivable — especially at this scale. Our hope is that these techniques may enable other researchers to gain access to similar types of data. Given the success of our methodology and the value we were able to generate from access to the configurations, perhaps more organizations can be led to anonymize and release their network configurations. The creation of a shared data-set of configurations would enable the direct clarification of many questions, such as network topology, that researchers have debated, and potentially increase network security for all concerned through an open-source-like review process.

4.1 Anonymizing Configuration Files

Our current anonymizer [17] is specific to Cisco IOS, but the strategy is generally applicable. All comments are removed from the configs using regular expressions. Under the assumption that no "secret" or identifying information would appear in the published Cisco IOS command reference guide, but that most valid IOS commands would be found there, all of the words found in the guide were extracted and turned into a list of tokens that that do not require anonymization. All non-numeric tokens in the configurations are checked against this list, and any tokens not found in the list are hashed using SHA1 digests [1]. This anonymizes the names of class-maps, route-maps, and any other strings that could hold privileged information. Simple integers are generally not anonymized. Regexps are used to locate and anonymize all public AS numbers, although private AS numbers are not hashed since they do not leak information about the identity of the network. All IP addresses are hashed using a modified version of the tcpdpriv algorithm [19].

While hypothetical attacks on the tcpdpriv algorithm have been proposed [28], they use the frequency with addresses appear in a packet trace — information that is not available from anonymized static configuration files. An attack on the IP address anonymization could be attempted by fingerprinting the pattern of address space usage inside a network and probing addresses in candidate networks to look for a match, but most of the networks in our data set filter the packets that would be needed to conduct the probing. Even if the network permitted probing, determining the number of /30s, /29s etc., from ICMP Reply or backscatter packets would be quite challenging. In our case, these anonymization techniques have proven sufficient to give our partner telecommunications company a sufficient level of comfort to grant us access to the configuration files for this study.

As a result of the anonymization process, all comments, documentation, semantically meaningful names, or anything that could convey intent is removed from the configurations. All that is left is raw mechanism. Since the configuration files for each network are placed in a single directory with filenames of the form "config1", "config2", ... not even DNS naming conventions that have been used by other groups to guess at POP structure and geographic location are available. The lack of comments and meta-data increases the challenge in reverse-engineering the routing design of the network, but highlights the value of the modeling techniques proposed in this paper to extract the structure of the routing design from an unorganized mass of configuration files.

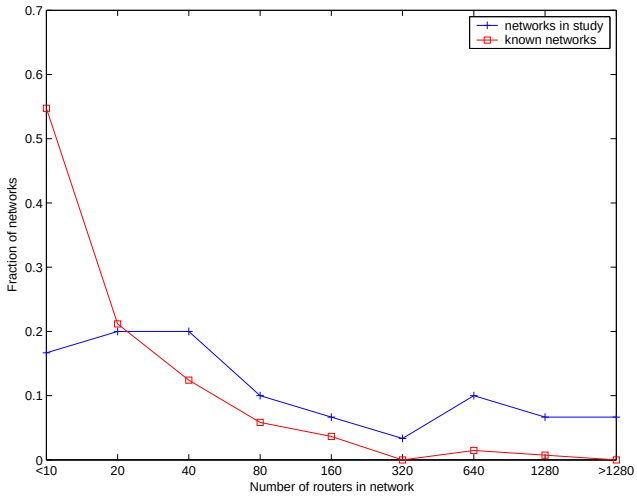

Figure 8: Distribution of the size of the 31 analyzed networks compared to the size distribution of all networks known in this repository.

4.2 Size of the Analyzed Networks

Figure 8 shows the distribution of the sizes of the 31 networks we studied in detail compared with the sizes of 2,400 networks in a repository available to us. Our study contains networks from across the range of network sizes found in the wild, with a slight overweighting towards networks with more than 20 routers.

5. Discoveries from Routing Design Analysis

Using tools built to embody our reverse engineering methodology, we conducted extensive analysis of 31 networks whose configurations were available to us. In this section we discuss some of the more interesting routing related observations that show the power of the analysis of static configuration files to frame and answer questions about networks that were previously unexamined.

5.1 Using the Routing Instance Model to Understand a Network's Structure

To illustrate how reverse engineering the routing design can take even an extremely complicated network and make it intelligble, we examine an enterprise network called net5. This network contains a total of 881 routers and 14 different BGP ASs, all internal to the network. There are 24 routing instances, which range in size from the largest that contains 445 routers to the smallest that contains only a single router. The network connects to external networks through EBGP sessions with 16 different external ASs.

A graph of the physical topology of net5 is an unintelligible "hairball" — a densely connected set of routers and links that gives no insight into how the network is structured or

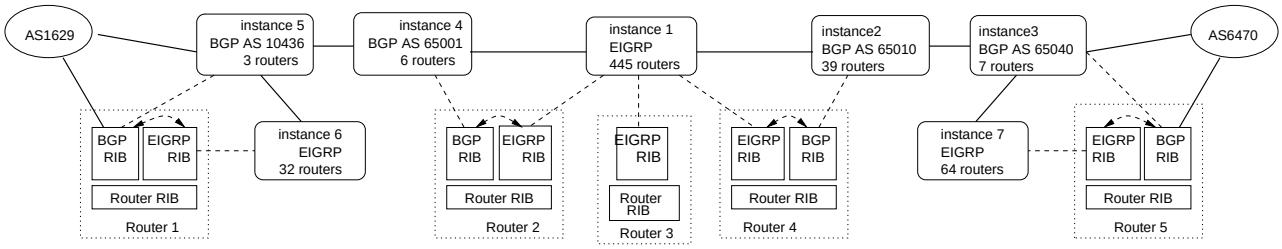

Figure 9: The routing design of three compartments in network net5. Net5 uses EIGRP as an inter-domain protocol to redistribute external routes between BGP instances 2 and 4, and EBGP as an intra-domain protocol to redistribute internal routes between BGP instances 2 and 3. Routes to external destinations learned by the 445 routers in EIGRP instance 1 have been passed through at least 3 layers of routing protocols and redistributions - all inside the single network.

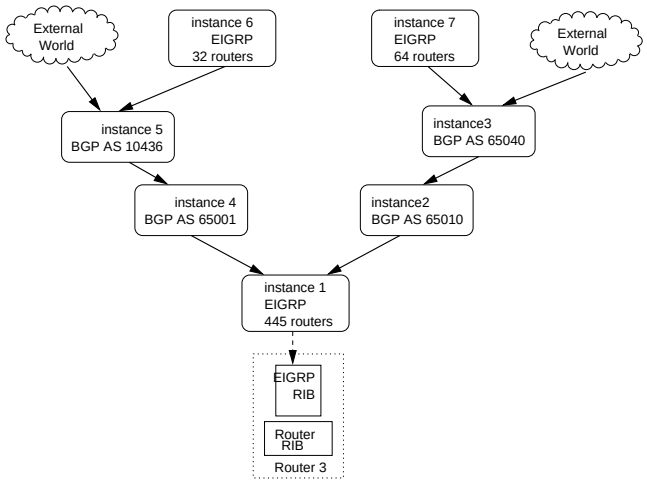

Figure 10: The route pathway graph by which Router 3 learns routing information.

functions. In contrast, Figure 9 shows the routing instance graph for the majority of net5 (541 out of the 881 routers).

As shown in the figure, most of the network's routers are connected to one of three routing instances running the EIGRP protocol: 445 routers to instance 1, 32 routers to instance 6, and 64 routers to instance 7. Routes are redistributed between the EIGRP instances by four different BGP instances, each with a different AS number. Using the routing instance diagram as a key, it is possible to navigate through the configuration files finding the exact set of routers whose configuration must be understood in order to answer a given question.

For example, a question might be "how many routers need to fail before instance 1 is partitioned from instance 2?" As shown in the routing instance graph, the function of router 2 in the routing design is to redistribute routes between routing instances 4 and 1. There are 6 routers in net5 that serve this same purpose, and they serve as redundant backups for each other. If all these 6 routers were to fail, these two instances would be separated (unless they were somehow reachable to each other through the external world, which is not true in this case)

Figure 10 shows the route pathway graph for router 3, which sits in the middle of net5. This figure can be used to answer questions such as "will packets sent to the outside world by router 3 use the egress point at the far left of the network, or the far right?" The route pathway graph shows which routers have policy that governs how routes are propagated to router3.

The route pathway graph for router 3 also makes clear that the routing design of net5 does not follow either the conventional enterprise or backbone network pattern. The use of routing protocols in net5 simply cannot be fit into the conventional model of a two layer EGP/IGP network.

5.2 IGP vs EGP classification

To quantify how many networks use an unconventional routing design, we examine how the networks make use of routing protocols. Routing protocols are commonly categorized as either Interior Gateway Protocol (IGP) or Exterior Gateway Protocols (EGP). The IGP and EGP classification refers to a protocol's routing scope in relation to an administrative domain. RIP, OSPF, IS-IS, and EIGRP have been cast as IGPs primarily because they lack the route selection attributes believed to be required of an EGP, and because of their historical application as protocols whose operation remained entirely inside an administrative domain. It is also widely accepted that BGP is the only EGP for IP networks.

One of the first observations we make is that the use of routing protocols in many of the 31 networks does not follow the IGP-EGP classification. To compute the frequency with which each routing protocol serves in a given role, we developed a method to classify the roles of all routing protocol instances employed by the 31 networks. Routing protocol instances that have adjacencies with the instances of another network are considered to be serving as an EGP or *inter-domain* protocols; otherwise they are being used as an IGP or *intra-domain* protocol.

Determining whether a routing protocol instance has an adjacency with a routing instance in another network is not easy, as many links in IP networks are multi-point and unless all the addresses in the link's subnet are found in the configuration files of other routers in the network, it is possible that an external router is present on the link - making the link an external peering point.

Point-to-point links commonly use a /30 subnet, which generally contains only 2 usable IP addresses. If both addresses are present in the configuration files, the interface is declared internal-facing, otherwise it is declared external-facing. Multipoint links, such as Ethernet, commonly have much larger subnets assigned to them (e.g., a /24 containing

Table 1: Number of protocol instances performing intra- or inter-domain routing.

	EBGP	IGP			
	Sessions	OSPF	EIGRP	RIP	Total
Intra-	1,490	9,624	12,741	156	22,521
Inter-	13,830	1,161	1,342	161	2,664

256 address). Such interfaces could be internal-facing and used to connect individual hosts, or they could implement a shared "DMZ" specifically designed to connect internal and external routers. However, routing configuration can be used to determine the true purpose of the link. If the interface is used as the next-hop for any external destinations (i.e., addresses not known to be inside the network as determined by analysis of the address space structure), then we assume there must be an external router on the link to accept and forward these packets and we mark the link as external-facing.

Table 1 shows both the number of different routing instances found in our 31 networks, broken out by routing protocol and intra/inter role in the network. In our data we saw no use of IS-IS. The number for intra-domain use of EIGRP includes two instances of IGRP. For OSPF, EIGRP and EBGP, the vast majority of sessions conform with conventional wisdom, with about 90% of OSPF and EIGRP sessions being used for intra-domain routing and about the same fraction of EBGP sessions being inter-domain. However, the data show a more diverse use, with a significant number of sessions breaking the conventional paradigm. Among the 31 networks, there are a total of 2664 cases (11% of total) where an IGP protocol instance performs the function of an EGP and 1490 instances (10% of total) where an EBGP session is used for intra-network routing. Three networks do not use BGP at all.

We hypothesize several reasons why designers use IGPs like RIP, OSPF, EIGRP, etc. as *edge* protocols that talk to another network: be it their provider or their customer. It is widely believed that these protocols are easier to configure than BGP, and there is anecdotal evidence that these protocols consume fewer memory resources than a BGP process (which is significant for low-end enterprise routers). There are also several reasons why EBGP might be chosen for use an *internal* protocol. Perhaps the network designer sought greater scalability by dividing the network into compartments, or perhaps the EBGP sessions are a legacy from when several separate networks were merged together to form a single network during a corporate merger. Alternatively, the use of BGP gives the network designer a fine degree of control over route selection and reachability with its extensive routing policy features.

5.3 Restricting Reachability in the Network

Another surprising result from our analysis is that some of the networks have many packet filters applied to the *internal* links. This type of reachability restriction inside the network has not been documented before. According to conventional wisdom, "protective" route and packet filtering are only required at the edge of a network to avoid bogus route advertisements and spoofed packets [6]. To evaluate how packet filtering is used in production networks, we gathered packet filter usage statistics for each network. Three networks do not have any packet filter definitions, and they are ignored for the purpose of this analysis reducing the data set size to 28 networks.

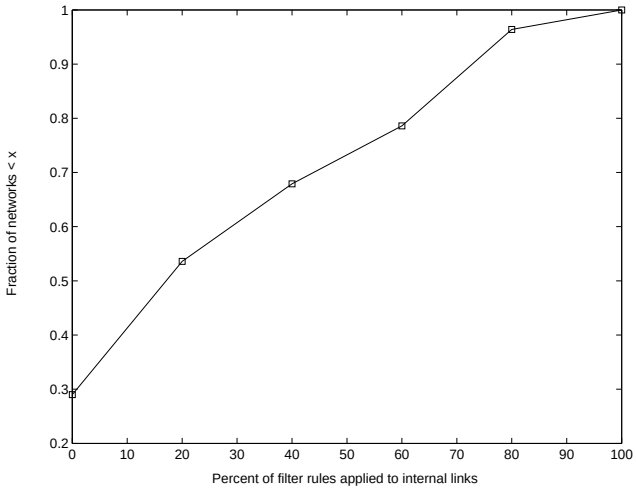

Figure 11: CDF of percentage of packet filters applied to internal links

The basic building block of a packet filter definition is an access control list or route-map that consists of a variable number of "if condition then action" clauses. To measure the total amount of filtering policy on a link, regardless of how the policy is grouped into filters, we treat each clause as a separate filter rule.

Figure 11 plots the cumulative density distribution of the percentage of packet filter rules applied to internal links for the 28 networks. The figure shows that in more than 30% of the networks, at least 40% of the packet filter rules are applied at internal interfaces.

We investigate further by examining the details of all the internal packet filters. The results show a great diversity in the policy goals that these filters try to achieve. For example: They are used to drop packets of a specific protocol (e.g., PIM) originating from an internal host, effectively disabling that protocol in all or parts of the network. They are used to block traffic that uses certain UDP or TCP ports. They are also used to dictate which set of hosts can use a particular application through selective filtering based the application's port.

Our detailed look at the packet filters also reveals weaknesses in the Cisco IOS language that can make configuring routers more error prone and the maintenance of router configurations more difficult. For example, we had a hard time deciphering the purpose of one packet filter because it consists of 47 clauses defining several policies simultaneously. A better design would be to create multiple packet filters, one for each policy. However, IOS only allows a single filter to be applied to each interface, thereby forcing the designer to place all 47 clauses into a single filter.

6. Case Studies of Routing Designs

From the 31 analyzed networks, it is very clear that the use of alternate architectures is not just a theoretical pos-

sibility, it is very common. Beyond even the unusual use of "interior" and "exterior" gateway protocols described in Section 5.2, we have found a vast array of different routing designs in use in these production networks. The diversity highlights the need to study and understand these designs. As a community, we must determine if they represent a clever optimization of previously unknown trade-offs, or if they are simple kludges and mis-designs that could be avoided if the research community develops a repository of best-common practices for routing design through the study of such positive and negative examples.

As a first step in this direction, this section presents a case study of the routing design of two networks to show how the designers' latent insights can be brought out. The first case study examines how the need for an IBGP mesh can be avoided, even in networks that must implement complicated routing policy. The second case study examines how reachability to external destinations can be controlled.

6.1 Avoiding an IBGP Mesh

Analysis of the routing design of net5 (Figure 9) shows that its highly unconventional structure is not a mistake, but rather a clever design optimized to the constraints of the network. For reasons we have been unable to discern from the configurations, the designer felt the need to segregate the network into compartments. This compartmentalization, combined with the large number of places the network peers with external networks, creates a large number of potential egress points for packets being routed in the middle of the network.

Under conventional wisdom, the need for complex route selection logic in the middle of the network to choose among egress points and the need to pass route information between network compartments should cause the designer to decide use a backbone-like routing design. In a backbone design (typified by router 5 in Figures 5 and 7), IBGP sessions are used to distribute external routes through out the network, because implementing complex route selection logic generally requires the use of route attributes carried by BGP, such as AS-path information, that would be lost if the routes were redistributed into an IGP.

However, the designer of net5 appears to have deliberately constructed the network to avoid the need for BGP attributes in route selection, enabling an IGP to be used for redistribution of all routes inside the compartments of the network. Two techniques were used to achieve this simplification.

First, the address blocks used in the network were carefully laid out so that the routers and hosts in each network compartment use addresses from an address block made up by a small number of unique and non-overlapping subnets. This meant that even complicated redistribution policies could be expressed using only address-based route-maps, as the address space was laid out to support the containment the designer was trying to achieve. There was therefore no need for the use of BGP-style attributes like AS-paths to control route redistribution.

Second, external routes were tagged to indicate their source as they were first redistributed into the network's IGP instances. Route selection for the router RIB on each router was configured to key off the tag, and since the IGP can propagate these tags, the need for an IBGP mesh and related BGP configuration was avoided.

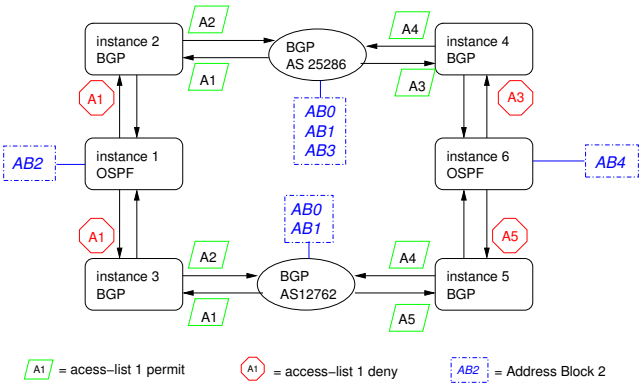

Figure 12: Routing design of network 15 annotated with policies.

Backbone networks do not have the freedom to assign addresses to all the peers whose packets will transit their network, so their routers must use AS-path attributes to decide which routes should be placed in their RIBs or redistributed. In creating net5's routing design, the network designer was able to avoid the need to configure an IBGP mesh that would distribute external routes throughout the network. Given the large number of nodes in the network, a simple IBGP mesh would not be scalable, and a complex set of IBGP reflectors would be required.

The example of net5 indicates that the space of trade-offs in which network design takes places is much larger than originally believed. By analysis of an operational network's routing design we found evidence supporting a new class of trade-offs: namely a tension between structured address assignment that enables simplified routing policies and arbitrary address assignment which requires more complex routing designs and routing policies.

The exercise also validates our goal that routing design extraction can be used to help understand and assess the structure of networks for which no documentation is available.

6.2 Controlling External Reachability

While the ultimate goal of networking is to enable communication between hosts that are not directly connected, in the networks we studied we observed a wide diversity of mechanisms being used to limit the set of destinations that hosts could reach. A completely accurate answer to the question of which hosts can communicate is extremely difficult, as it requires modeling the details of the route selection algorithm used by each protocol on each router on the network. However, by applying routing design analysis there is a middle ground that avoids the need to model route selection but still provides an extremely useful view of the reachability provided by the network.

We have developed a reachability analysis algorithm [27] that begins with the routing instances calculated as described in Section 3. Figure 12 shows an example of this analysis applied to net15, which has 79 routers. The network connection to the outside world is via EBGP peering sessions with two different public ASs (anonymized as AS 25286 and AS 12762). The network has six routing instances, shown with rounded boxes, and the links connecting them repre-

Table 2: Address blocks mentioned by redistribution policies.

Policy #	Contents	Policy #	Contents
A1	AB0, AB1	A3	AB0, AB3
A2	AB2	A4	AB4
		A5	AB0

sent the route redistribution between the instances. Where there is policy restricting the redistribution, the link is annotated with a policy number and whether the routes specified by the policy are blocked (stop sign) or permitted (flag). The subnets mentioned in the policies have been aggregated into numbered Address Blocks, which are connected to the routing instances where those subnets are attached to the network. From this figure and Table 2, which shows which address blocks are mentioned in each policy, the following observations emerge:

First, this is an example of a network where hosts do not have reachability to the Internet at large. The only routes allowed into the network from the public BGP ASs are those listed by policies A1, A3, and A5, which total two /16 networks and 3 /24s. There is no default route permitted. However, routes to the hosts connected to the network (address blocks AB2 and AB4) are allowed out. It is impossible to tell whether the public ASs propagate these routes further, but from a security standpoint it is possible that packets from the global Internet will find their way into this network, although the hosts inside the network will not be able to respond.

Second, from the structure of the routing design, presumably the routers in instances 1, 2, and 3 are located in a different location from those in instances 4, 5, and 6. However, the connections to the public ASs are not used to create a virtual private network between the two sites. In fact, packets from hosts connected in Address Block 2 cannot reach hosts in Address Block 4 at all, or vice versa. The intersection of the policies that control routes leaving the left half the network with those controlling routes entering the right half of the network is the null set: $A2 \cap A5 = A2 \cap A3 = A4 \cap A1 = \emptyset$.

Third, as is common in enterprise networks, OSPF is used to redistribute routes inside the network. However, by using the routing instance model, it is now possible to begin predicting the scalability of the routing design. The reachability analysis establishes that the ingress filters A1, A3 and A5 are the factors that control the maximum number of external routes that can be injected into the OSPF instances. Combined with the number of routers in the OSPF instance, the maximum load on the OSPF processes can be predicted.

7. Differences Among Routing Designs

Networks are designed and built to serve a purpose, such as connecting together the computers used by a corporation or creating a transit backbone. In other fields, commonality of purpose has led to commonality in the mechanisms used to achieve these purposes through the construction of "cookbooks" or "best practices." In routing design, the classic textbooks [12] generally define only two architectures: the enterprise and backbone architectures presented in Section 3. In this section, we evaluate how closely production networks follow these architectures, and we verify whether heuristics commonly used to classify networks are accurate or not. Our data set of 31 networks contains 4 backbone networks, enterprise networks of varying sizes, and several large tier-2 ISPs.

7.1 Routing Design

Although enterprise networks and backbone networks are commonly thought of as following canonical "textbook" architectures, in reality the routing designs used by networks are far more diverse than these two architectures.

The four backbone networks do exhibit features close to the textbook backbone routing design: a large number of EBGP sessions are used to peer with external networks; IBGP is used for distribution of external routes from the border routers to interior routers; and a small number of IGP instances is used to distribute routes to internal subnets.

The large tier-2 ISP has the BGP structure of a backbone network, but contains a very large number of *staging IGP instances.* These are routing instances of a traditional IGP protocol, like OSPF or EIGRP, that have only a single router inside the network, but a large number of external peers. Presumably these are used to connect customers that do not run BGP to the tier-2 ISP. Instead, an IGP protocol is used to distribute routing information to the customers. Network designers tell us that this is often done in preference to using static routes because the IGP provides ongoing validation that the link to the customer is still up.

Seven of the 31 networks had routing designs very close to a textbook enterprise network: a small number of BGP speakers that communicate with the outside world and inject routes into a small number of IGP instances from which most of the network's routers learn their routes. The largest of the seven divided up its 101 routers equally between two IGP instances, presumably to improve performance and scalability.

The remaining 20 enterprise networks exhibited designs that were so markedly different both from textbook examples and from each other as to defy classification. Figure 7 shows how routes are redistributed between IGP and EGP for the canonical backbone and enterprise architectures using route pathway graphs. The difference between this figure and Figure 10 showing the route pathway graph for net5 makes clear how different production routing designs can be. The hierarchy of protocols and route redistribution used in net5's routing design is only one example of the many different structures observed across the 31 networks we examined. 17 of the networks involved some form of the redistribution of external routes learned via BGP into an IGP, but they differed in the number of ASs used internal to the network, the arrangement of those ASs, the completeness of the IBGP mesh inside the ASs, and the redistribution of routes between the ASs.

7.2 Size

Backbone networks are commonly assumed to be the largest networks, but network size is not a good indication of network type. The four networks with a backbone architecture range in size from 400 to 600 routers, with a mean of 540 routers. The seven networks with a classic enterprise structure tend to be quite small, with sizes ranging from 19 to 101

Table 3: Types of interfaces found among the 31 networks in analyzed data set.

Type	Count	Type	Count
Null	2	Dialer	1296
Multilink	4	TokenRing	1344
Fddi	6	GigabitEthernet	2171
CBR	14	Hssi	2375
Channel	51	Ethernet	3685
Virtual	83	POS	3937
Async	90	ATM	6242
Port	151	FastEthernet	20420
Tunnel	202	Serial	53337
BRI	1077		
		total # :	96487

routers. In contrast, the 20 enterprise networks with an unclassifiable routing design vary in size from 4 to 1750 routers, with a mean of 300 and a median of 36. As shown in Figure 8, the distribution is skewed towards smaller networks, but there are still four networks larger than the largest backbone network, containing 760, 890, 1430, and 1750 routers. These networks belong to large enterprises and tier-2 ISPs.

7.3 Interface Composition

The interfaces used in a network are a relatively good predictor of the type of the network. As expected, three of four backbones are largely built using Packet-Over-SONET (POS) interfaces, which are high-cost interfaces normally reserved for high-speed long-haul links. Table 3 shows the type and frequency of the interfaces in use on the 8,035 devices in the 31 networks we studied. By far, serial interfaces are the most common. The links connected to these interfaces are most commonly implemented using private T1 lines or frame-relay circuits. Ten of the networks make extensive use of ATM as a layer-2 technology for connecting their routers. While POS interfaces are heavily used in three of the four backbone networks, they appear in two enterprise networks as well. The fourth backbone is based on High Speed Serial Interfaces (HSSI) and ATM.

8. Routing Design in a Larger Context

Our analysis of router configuration data sheds light on the poorly-understood art of routing design. In this section, we discuss why configuration data alone does not provide a complete view of the designers' intent. Still, our model of the routing design can serve as an important building block supporting important tasks in running a large IP network.

8.1 Routing Design Data as a Building Block

In practice, the design and operation of large IP networks consists of wide variety of tasks on different timescales. Having an accurate, up-to-date view of the routing design — constructed from configuration files or from an external database — is extremely useful for conducting these tasks:

Inventory management: Maintaining an up-to-date view of the network equipment, router configuration, and the assignment of IP address blocks is an important part of running a large IP network. The routing design extracted from the configuration files is valuable in checking the consistency of external inventory databases, or to provide new inputs to the database after acquiring a new network. Snapshots of the routing design over time can be used to track the steps in adding or removing equipment from the network. The network designer can also use the current routing design to determine where and how to add a new link or router to the network, including the planning of which routing protocol adjacencies and parameters to configure.

Vulnerability assessment: The routing design model provides a concise summary of the routing protocols and parameters used in the operational network. This information can be used to assess potential network vulnerabilities, such as vulnerability to network attacks, configuration errors, or violations of *best common practices*. For example, the operator can identify connections to neighboring domains that do not have packet or route filters, or internal links and routers with incomplete routing protocol adjacencies.

Network engineering: The operators can also evaluate the robustness of the routing design to equipment failures and planned maintenance activities. For example, analysis of the routing design data can uncover scenarios where a single link or session failure would disconnect part of the network. The operators can also schedule maintenance activities to avoid disabling multiple routers with static routes to the same destination prefix, to limit the likelihood of service disruptions. In addition to the routing design data, survivability analysis requires "what if" tools that model the effects of changes to the network topology and routing configuration [5]. The analysis may also require additional information about the mapping of IP links to layer-two (e.g., ATM switches) and layer-1 (e.g., fiber spans and optical amplifiers) to accurately capture the effects of failures and maintenance activities in the underlying transport network. With accurate measurements of the offered traffic, the operators can use the "what if" tools to determine the effects of changes to the routing configuration and the underlying topology on the traffic load in the network.

Anomaly detection and diagnosis: Detecting and diagnosing anomalous behavior is a crucial part of running a large IP network [3]. Operators need effective ways to identify why a user cannot reach a particular destination, or why a routing protocol is flapping. Ultimately, anomaly detection depends on analyzing a wide range of network measurements of link and CPU load, packet and flow traces, performance statistics, fault alarms, and routing protocol messages. Diagnosing a problem may require the operator to probe the network using tools such as ping and traceroute. However, making sense of these data sets requires information about the routing design. For example, the routing design captures the fact that an EBGP session is associated with a particular edge interface, which may be important in explaining why the BGP session has failed. The routing design also reveals situations where two hosts should *not* be able to reach each other, due to packet or route filtering policies. Finally, the routing design is crucial for deciding *where* to place the measurement devices (such as packet monitors or routing monitors) to collect the most useful data.

In practice, an accurate, up-to-date view of the network topology, routing protocol configuration, and packet/route filters is crucial for supporting these and many other network management tasks.

8.2 Challenges of Inferring the Routing Design

Certain aspects of routing design are not captured in a snapshot of a network's configuration state:

Uncovering the reasons for the design decisions: The analysis of our data reveals the network topology and routing protocol configuration but does not explain the reasoning behind these choices. For example, many large enterprises have a hub-and-spoke topology to provide spokes (e.g., retail stores) access to hubs (e.g., central places for maintaining inventory and credit-card charges). However, this does not imply that the spokes never communicate with each other, routing through the hub to do so. In some cases, the hub-and-spoke design might be motivated in part by cost constraints. In other cases, the network designer might not know that the spokes do indeed communicate (through the hub). Spoke-to-spoke communication would only be visible by measuring and analyzing traffic data. Similarly, a hierarchical routing design might suggest that a network consists of separate administrative regions; e.g., with different autonomous systems administered by different operations teams. However, the same design might arise to bound the processing load on the control plane of the routers within each autonomous system. In some cases, a more detailed analysis might hint that the explanation indeed lies in partitioning administrative responsibilities (e.g., presence of different passwords, operating system versions, or patterns of configuration commands). Still, understanding the true intent of the designer(s) is difficult without more information.

Absence of important side information: The configuration files do not include basic information such as physical locations or the distances between the routers. Depending on the interface technology, the capacity of a link may be unknown or dependent on the underlying layer-two circuit. In some cases, important information may be gleaned from knowing the network operators' conventions for naming the routers, entering comment strings, or assigning tunable parameters. For example, the hostname of a router might implicitly indicate its location, vendor, model, and role in the network. DNS names associated with routers and interfaces are sometimes used in a similar fashion. Comment fields in the interface section might indicate the role of the link (e.g., connection to a customer or peer), the name of the neighbor, and whether the interface is in the middle of provisioning. Specific values of routing protocol parameters (such as OSPF link costs) might indicate the type of link (e.g., intra-PoP or inter-PoP) and whether the link is undergoing maintenance. Acquiring this kind of "side information" from network databases and operators, while challenging and sometimes error prone, is extremely worthwhile because it makes newer, deeper forms of analysis possible.

Limited information about the neighboring domains: The router configuration files only provide information about one end of the links and sessions to neighboring domains. Although the packet and route filters on edge links constrain the behavior of the neighbors, reasoning about the expected or typical behavior is challenging. In the extreme, an edge link might not have any filtering at all, making it impossible to know what kinds of data packets and route advertisements to expect. This problem becomes much simpler if packet traces, routing table snapshots, or the configuration files for the remote routers are available. In addition, the configuration data does not reveal whether the routers inside the network can communicate with each other through neighboring domains. For example, a network with links to two neighboring domains may have a "backdoor" route through these external connections. The presence of such backdoor routes is difficult to discern, even for the network operators themselves. Often, routing table dumps or traceroute data are necessary to uncover these kinds of situations.

Evolution of the routing design over time: In practice, routing design is not a discrete activity that takes place a single time when a network is first built. Instead, design is a continual process. At any given time, a network may have elements of old and new designs, including vestiges of incomplete or abandoned modifications to the configuration. Similarly, the provisioning and decommissioning of equipment may lead to network configurations that appear incomplete or inconsistent. In addition, mergers and acquisitions may lead to hybrid designs with distinct characteristics that date back to the original designs of the constituent networks. For example, a single network might use OSPF as the IGP in certain domains and EIGRP in others for purely historical reasons. Acquiring a deeper understanding of the evolution of the routing design requires a longitudinal analysis with multiple snapshots of the router configuration data over time. We plan to pursue this analysis as part of our ongoing work.

9. Related Work

In the absence of the data needed to conduct white-box analysis of routing designs, there has been significant work on black-box reverse engineering of network topology and IP connectivity [25, 2, 8]. The work of [5, 3] illustrates the potential power of white-box network analysis, via automated processing of router configuration files. Many network management tools for network and traffic engineering often rely on similar methods to obtain topology and routing configuration information [14, 5]. A wealth of data on routing behavior has been gleaned from routing table dumps and route monitors, particularly the BGP data collected by the RouteViews project [18]. Such techniques, deployed within a given routing domain [24, 15], provide dynamic white-box measurements of IP connectivity and reachability information. Though such data would complement and enhance the investigation considered here, the associated instrumentation has not yet been widely deployed.

We considered existing data models, but none were appropriate for modeling routing designs. ITU-T M-series recommendations [23] are more geared for inventory management. The Distributed Management Task Force (DMTF) has created a model for representing the configuration of networks [11], but it is at the wrong granularity for the study of routing design. It provides no means of abstraction, like our routing instances, or means of analysis, like our route pathway and routing process graphs.

10. Summary

An IP network's routing design is embodied in the configuration of its routing protocols. Through the routing design, network operators attempt to balance complex objective and constraints, and to ensure robust network operations. In this paper, we make three primary contributions:

1. We present a methodology for working with the configuration files of production networks that supports the reverse engineering of the network's routing design.

2. We provide a general method to (i) automatically process the configuration files for a given network to extract the primitives that make up the network's routing configuration and populate a router level model of the network, and (ii) derive coherent global views of the network's routing design from the individual primitives spread across the configuration files. These global views include the *routing process graph*, *routing instance graph*, *route pathway graph* and *address space structure*. Together, they provide a means to abstract and summarize a network's configuration that exposes the structure of the routing design and opens it up to direct analysis.

3. We demonstrate the value of our approach by presenting examples of the application of the techniques to thirty-one production networks, and 8,035 configuration files. This structural information is essential for several important operational tasks: inventory management; vulnerability assessment; network engineering; anomaly detection and diagnosis.

Some of the unconventional features of the routing designs illustrate difficulties in meeting complex objectives and constraints in operational networks. Others illustrate that, like programming, routing design is an art where many approaches might be used to try to achieve the same result. We believe that the best way forward for the operational and research communities to improve routing designs is to first understand the details, strengths and weaknesses of existing designs.

We see our methodology as part of series of steps towards a holistic theory of the design and operation of data networks. Understanding the the mechanisms and dynamic behavior of individual routing protocols is insufficient.We must work towards a framework for understanding the interactions between the individual protocols and mechanisms from which the network is forged in order to make progress on the goal of achieving more scalable and robust networks.

11. Acknowledgments

We are grateful for the significant contributions Jennifer Rexford has made in advancing this work and revising this paper.

12. REFERENCES

[1] D. Eastlake 3rd and P. Jones. *RFC 3174 - US Secure Hash Algorithm 1 (SHA1)*, 2001. Available from http://www.ietf.org/rfcs/rfc3174.html.

[2] CAIDA. http://www.caida.org/tools/measurement/skitter/, 2000.

[3] Don Caldwell, Anna Gilbert, Joel Gottlieb, Albert Greenberg, Gisli Hjalmtysson, and Jennifer Rexford. The cutting EDGE of IP router configuration. In *Second Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[4] R. Callon. *RFC 1195 - Use of OSI IS-IS for routing in TCP/IP and dual environments*, 1990.

[5] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, and Jennifer Rexford. Netscope: Traffic engineering for IP networks. *IEEE Network Magazine*, pages 11–19, March 2000.

[6] P. Ferguson and D. Senie. *Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing*. Internet Engineering Task Force, January 1998. RFC 2267.

[7] Lixin Gao and Feng Wang. The extent of AS path inflation by routing policies. In *Proceedings of Global Internet 2002*, 2002.

[8] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for internet map discovery. In *IEEE INFOCOM 2000*, pages 1371–1380, Tel Aviv, Israel, March 2000. IEEE.

[9] Timothy Griffin, F. Bruce Shepherd, and Gordon T. Wilfong. Policy disputes in path-vector protocols. In *Proceedings of the 7th Annual International Conference on Network Protocols*, pages 21–30, Toronto, Canada, November 1999.

[10] Timothy G. Griffin and Gordon T. Wilfong. An analysis of BGP convergence properties. In *Proceedings of SIGCOMM*, pages 277–288, Cambridge, MA, August 1999.

[11] DMTF Networks Working Group. http://www.dmtf.org/standards/cim/cim_schema_v27.

[12] Sam Halabi and Danny McPherson. *Internet Routing Architectures*. Cisco Press, 2001.

[13] C. Hedrick. *RFC 1058 - Routing Information Protocol*, 1988.

[14] OPNET Technologies Inc. http://www.mil3.com/products/home.html.

[15] Packet Design Inc. http://www.packetdesign.com.

[16] D. E. Knuth. An empirical study of FORTRAN programs. *Software - Practice and Experience*, 1(2):105–133, April-June 1971.

[17] David A. Maltz, Jibin Zhan, Geoffrey Xie, Hui Zhang, Gisli Hjalmtysson, Albert Greenberg, and Jennifer Rexford. Structure preserving anonymization of router configuration data. Technical Report CMU-CS-04-149, Carnegie Mellon University, 2004.

[18] David Meyer and University of Oregon Route Views Project. http://antc.uoregon.edu/route-views/.

[19] Greg Minshall. tcpdpriv - remove private information from a tcpdump -w file. Software distribution available from http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html, 1997.

[20] J. Moy. *RFC 2178 - OSPF Version 2*, 1997.

[21] Vern Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, 1997.

[22] Y. Rekhter and T. Li. *RFC 1771 - A Border Gateway Protocol 4 (BGP-4)*, 1995.

[23] ITU-T M series recommendations. http://www.itu.int/rec/recommendation.asp-?type=products&lang=e&parent=T-REC-M.

[24] A. Shaikh, L. Kalampoukas, R. Dube, and A. Varma. Routing stability in congested networks: Experimentation and analysis. In *Proc. ACM SIGCOMM'00*, pages 163–174, Stockholm, Sweden, 2000.

[25] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *Proc. ACM SIGCOMM*, August 2002.

[26] Cisco Systems. *Enhanced IGRP*. http://www.cisco.com/-univercd/cc/td/doc/cisintwk/ito_doc/en_igrp.htm.

[27] Geoffrey Xie, Jibin Zhan, David A. Maltz, Hui Zhang, Albert Greenberg, Gisli Hjalmtysson, and Jennifer Rexford. On static reachability analysis of IP networks. Technical Report CMU-CS-04-146, Carnegie Mellon University, 2004.

[28] Tatu Ylonen. Thoughts on how to mount an attack on tcpdpriv's "-a50" option... Web White Paper available from http://ita.ee.lbl.gov/html/contrib/attack50/attack50.html.