Automated Provisioning of BGP Customers

Joel Gottlieb, Albert Greenberg, Jennifer Rexford, and Jia Wang
AT&T Labs – Research; Florham Park, NJ
{joel,albert,jrex,jiawang}@research.att.com

Abstract

Manual configuration of IP routers is an expensive, time-consuming, and error prone process. For large Internet service providers, establishing service for new customers is a major part of the financial cost of running the network. Increasingly, these customers want to exchange routing information with their provider(s) using the Border Gateway Protocol (BGP), a complex and highly-programmable interdomain routing protocol. This paper describes how a provider can configure its connections to BGP-speaking customers, from the technical questions asked of new customers to the individual configuration commands to applied to the production routers. We present a case study of a Technical Questionnaire and describe how the provider assigns unique identifiers such as IP address blocks, interface names, and access-control list numbers on behalf of the customer. Next we describe an example set of provisioning rules that use the customer-specific information to generate a sequence of configuration commands—a "configlet"—for adding the new connection to the network; our configuration rules are expressed using Cisco Internet Operating System (IOS) commands as an example. Then we describe a database schema for storing and accessing the customer-specific data, and discuss how to use a virtual view on this database to populate a template that captures the syntax of the router commands. Our provisioning system provides an inexpensive, efficient, and accurate way for a provider to configure connections to new BGP-speaking customers.

1 Introduction

The Internet consists of thousands of Autonomous Systems (ASes) operated by diverse institutions ranging from college campuses and corporate networks to large Internet Service Providers (ISPs). Routers in different ASes use the Border Gateway Protocol (BGP) [1, 2] to exchange information about how to reach destinations throughout the Internet—in essence, BGP is the "glue" that holds the disparate parts of the Internet together. As such, local mistakes in configuring BGP-speaking routers can have global consequences, such as lost connectivity, forwarding loops, route oscillation, and surges in routing table size and processing load. To make matters worse, BGP is a complex and poorly-understood protocol that is highly configurable. Manual configuration, while time-consuming and error-prone, is extremely common, and may be responsible for many of the outages and anomalies that occur in practice [3]. This paper provides an operational perspective on how BGP is configured in an ISP network, and describes a system for configuring the routers automatically based on customer-specific data and the provider's policies for establishing service for new customers.

Many customers do not need to run BGP, or any dynamic routing protocol, on their connections to their upstream providers. Static routing suffices for many corporate and university networks. However, these kinds of networks increasingly use BGP to exchange routing information with multiple upstream providers, or with other local networks, because BGP gives them more control over how traffic enters and leaves their domains.

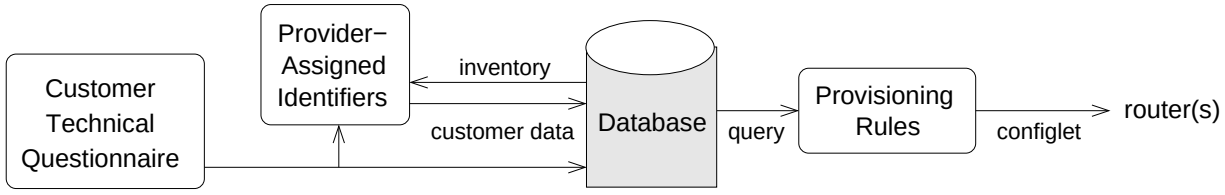


Figure 1: End-to-end view of the customer provisioning process

Even a network that connects to a single neighboring domain may use BGP to balance load over multiple links to the upstream provider. In practice, a customer network has a small number of BGP sessions that could be configured manually. On the other hand, a service provider typically manages a multitude of links and BGP sessions with a large number of neighboring domains. Adding new customers to the network is a major part of operating a large ISP, and these customers increasingly want to speak BGP. Establishing service for a new BGP-speaking customer involves configuring the access links and the associated BGP sessions, and assigning import and export policies to these sessions. The details of the configuration depend on customer-specific information such as the AS number, the destination prefixes, the number of access links, and the selected routing policies.

Automating the configuration of a provider's connection to a BGP-speaking customer argues for an end-to-end view from the initial data collected from the customer to the application of commands on the production routers. The key elements of our approach are illustrated in Figure 1. The Technical Questionnaire (TQ) lists the questions asked of a new customer. The responses to these questions flow through to a customer database. In addition, the provider may consult the database to assign unique identifiers (such as IP address blocks, interface names, and access-control list numbers) for the new customer. The provider has provisioning rules that determine how the customer-specific data affect the configuration of the routers. These rules drive the querying of the database and the generation of a list of configuration commands—a "configlet"—for adding the new customer to the network. Then, a separate system can connect to the production routers to apply these commands. In this paper, we focus on the TQ, the database, and the provisioning rules necessary to generate the configlet; we do not elaborate on the systems issues involved in applying the commands to the routers.

Establishing a clean separation between the customer-specific data and the provider's configuration rules is the main technical challenge in automating the provisioning process. In Section 2, we present an overview of BGP that highlights the requirements for configuring the provider's end of a BGP session with a customer. Then, Section 3 identifies the key customer-specific information that drives the provisioning process and describes how a provider acquires or generates this data. We also present a case study with example provisioning rules that take customer-specific data as input and generate a configlet as output. Our explanations draw on the syntax and semantics of Cisco Internet Operating System (IOS) commands as an example; other router vendors emulate the IOS command set or have configuration languages with similar constructs. Section 4 describes how to construct a database-driven provisioning system. We present a schema for representing the customer-specific data and discuss how to construct a virtual view that captures the customer's BGP configuration after the application of the provisioning rules. The paper concludes in Section 5 with a summary of our contributions

and a discussion of future research directions.

2 Border Gateway Protocol (BGP)

This section presents background material on BGP and highlights the requirements and parameters involved in configuring a BGP session. After a brief overview of Internet routing, we explain the steps in establishing a BGP session and discuss how a router knows how to reach the remote end of the session. Then we explain how BGP import and export policies affect the selection of the "best" route for each destination prefix and the propagation of routes to neighboring ASes.

2.1 Internet Routing

Packet forwarding in the Internet operates at the level of address blocks, or prefixes. Each prefix consists of a 32-bit address and a mask length. For example, $192.0.2.0/24$ represents the 256 addresses ranging from $192.0.2.0$ to $192.0.2.255$. An IP router constructs a forwarding table that associates each destination prefix with one or more of its outgoing interfaces. When a packet arrives, the router extracts the destination address from the IP header and finds the longest matching prefix; for example, the destination 192.0.2.179 could have a longest prefix match of 192.0.2.0/24. Then, the router forwards the packet to an outgoing interface based on the forwarding-table entry for this prefix. In the simplest case, a router has a forwarding table with a single, default route—for the prefix 0.0.0.0/0 that includes all IP addresses. Alternatively, the router may have a small number of statically-configured routes that map certain prefixes to particular outgoing interfaces. More generally, though, a router may construct its forwarding table by combining information learned from other routers via dynamic routing protocols.

The routers participate in an intradomain routing protocol, such as OSPF or IS-IS, to select paths within the AS. The routers at the edge of the domain can participate in BGP to exchange information with other ASes about how to reach destination prefixes. BGP is a path-vector protocol that constructs a route by successively propagating reachability information [2, 4, 5]. A BGP-speaking router sends an advertisement to notify its neighbor of a new route to the prefix and sends a withdrawal to revoke the route when it is no longer available. Each advertisement includes the attributes of the route, including the list of ASes along the path. For example, the AS path "7018 701" goes from AS 7018 (AT&T) to AS 701 (UUNET). Before accepting an advertisement, the receiving router checks for the presence of its own AS number in the AS path to detect and discard routes with loops. BGP is a policy-oriented protocol. Locally-configurable policies determine how a router selects the "best" route to each destination prefix (among the routes learned from its various neighbors) and whether to propagate the route to other neighbors.

2.2 BGP Session

Two routers exchange BGP messages over an underlying BGP session. Establishing the BGP session involves three main steps:

1. **Establishing the TCP connection:** The two routers establish a TCP connection between the two end-point IP addresses; in practice, each router is configured in advance with the IP address of the other BGP end point.

2. **Opening the BGP session:** Next the routers exchange BGP control messages to open the session. Each BGP speaker provides local information such as the BGP version number and the AS number.

3. **Exchanging routing tables:** Then each router advertises its "best" route for each destination prefix; each end of the session may have an export policy that restricts whether or not the route is propagated.

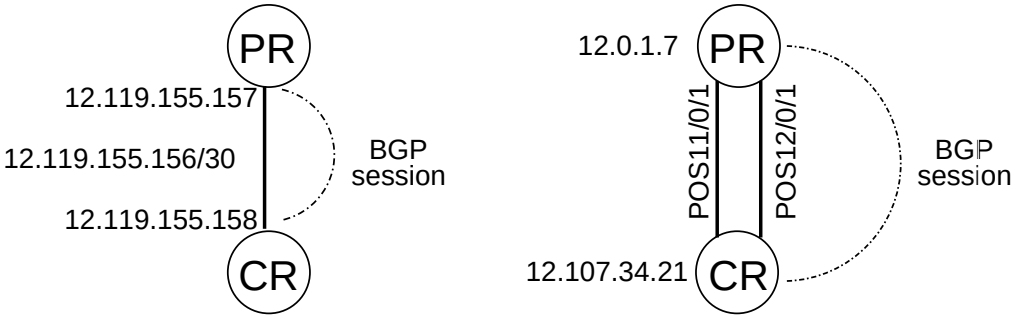
Finally, in steady state, the routers exchange update messages (advertisements and withdrawals) as the best routes change. The two routers also exchange periodic "keep-alive" messages to detect whether they can still communicate; the spacing between these messages is controlled by a configurable keep-alive timer. Each end-point has a configurable hold-time that is the maximum amount of time that may elapse between the receipt of successive keep-alive or update messages. If the hold timer expires, the BGP speaker returns to the idle state, waiting for an opportunity to establish a new BGP session. In the meantime, the router discards the routes learned via this session, selects new best routes where necessary, and sends update messages to notify its other neighbors of any routing changes.

To exchange BGP messages, each router must know how to direct IP packets to the remote end-point of the BGP session. This introduces a potential circularity—the routers must know how to reach each other in order to exchange routing information. Each router needs to know which outgoing link(s) would lead the packets to the remote end of the BGP session. There are three main ways for the Provider Router (PR) to reach the end-point of the BGP session on the Customer Router (CR):

- **Shared physical network:** The two routers may have a common physical network, as shown in Figure 2(a). In the prefix 12.119.155.156/30, the addresses 12.119.155.157 and 12.119.155.158 identify the two ends of the bidirectional, point-to-point link[1]. The PR uses the interface with address 12.119.155.157 to reach the remote end-point at 12.119.155.158.

- **Static configuration:** The PR could be configured to use certain outgoing interfaces to reach the other end of the BGP session. In Figure 2(b), the remote end-point is the "loopback" address (12.107.34.21) of the processor on the CR. The PR is statically configured to forward traffic destined to 12.107.34.21/32 via the interfaces POS11/0/1 and POS12/0/1.

[1] A point-to-point link typically has four IP addresses instead of two to allow for a network address (12.119.155.156) and a broadcast address (12.119.155.159).



(a) Directly−connected interfaces (b) Loopback interfaces

Figure 2: Two ways to reach the Customer Router's end of the BGP session

- **Dynamic routing:** In some cases, the PR may learn how to reach the IP address of the CR's end of the session via a dynamic routing protocol—including BGP itself. For example, a PR that has a BGP session with the RouteViews monitor (with IP address 198.32.162.100) might learn a route to 198.32.0.0/16 via some other BGP session.

The first two approaches have the advantage that the outgoing interfaces are known in advance and do not depend on dynamic information. In addition, these interfaces can be configured with packet filtering and resource allocation policies based on prior knowledge of what kinds of traffic should be traversing the links. The third approach is less common for sessions with normal customers that have one or more dedicated links to the provider; the operator might choose to configure such sessions manually.

Unlike most IP packets, the packets containing BGP messages usually do not traverse multiple links in the Internet. In fact, in the configuration in Figure 2(a), each BGP message travels a single hop from the sending router to the opposite end of the outgoing link. The router can give these IP packets an initial Time-To-Live (TTL) value of 1. This ensures that these packets do not mistakenly travel over a long path (say, in a forwarding loop) before the TTL value expires. However, in some cases the router may need to be configured to use a larger TTL value. For example, the packet may travel two hops when the routers communicate via their loopback addresses, as in Figure 2(b), or when the routers are separated by another network element such as a firewall or Network Address Translator (NAT). In some cases, the two routers could be separated by a large number of hops. For example, a routing monitor such as the RouteViews or RIPE-NCC collectors may have BGP sessions with operational routers in distant geographic locations. These kinds of BGP sessions would need to be configured with much larger TTL values.

2.3 Selecting and Propagating Routes

BGP is a policy-oriented protocol that gives operators a large amount of control over how the routers select and propagate routes. The handling of BGP update messages at a router is a three-step process:

- **Import policy:** Each BGP session at each router can be configured with an import policy that filters unwanted routes and manipulates the attributes of the remaining routes. Import filtering is a form of "defensive programming" that protects the network from erroneous BGP advertisements. For example, an import policy could discard all routes for so-called "martian" addresses (e.g., private IP addresses in the 10.0.0.0/8 block) that should not be announced in the Internet [6] or routes with AS paths that include private AS numbers in the range of 64512–65535 [7]. Attribute manipulation is a way to influence the selection of the "best" route for each prefix. For example, the import policy could assign a "local preference" to a route to override the default of selecting a route with the shortest AS path.

- **Decision process:** The router applies the BGP decision process to select exactly one "best" route for each prefix among the routes learned from each of the neighbors. This process consists of a long, relatively complex sequence of rules that rank the routes based on their many attributes, such as local preference, AS path length, and multiple-exit discriminator (MED). The attributes affecting the decision process are associated with the BGP route advertisements or may be assigned by the import policy. For example, a router would prefer a route with three-hop AS path and a local-preference of 100 over a route with a two-hop AS path and a local-preference of 90.

- **Export policy:** After selecting a single best route, the router applies an export policy for each session to decide whether to propagate the route and to manipulate the attributes of the remaining routes. Export filtering is a form of route "scoping" that may depend on the commercial relationship with the neighboring AS. For example, the BGP session between two "peer" ASes may have export policies that discard routes learned via other large providers to prevent transit traffic from traversing the two networks [8]. Export filtering also helps limit the size of routing tables in the Internet. For example, an AS may suppress routes for smaller subnets of larger address blocks, such as routes for prefixes allocated to customers out of the provider's own address space [9], particularly if the customers do not connect to any other service providers.

Router vendors offer flexible languages for operators to specify the import and export policies associated BGP sessions. The flexibility allows operators to apply diverse policies for discarding unwanted routes, influencing the selection of the best route for a destination prefix, and limiting the propagation of routing information to neighboring domains. In practice, a provider can define a small collection of import and export policies and install the definitions on the router in advance of provisioning any BGP sessions.

3 BGP Customer Provisioning

This section describes how to configure a provider router (PR) with a BGP session to a new customer. We first describe the customer-specific data required for the provisioning process and explain how the provider acquires this information. Then we present a case study with an example set of provisioning rules for configuring the

access interfaces, establishing the BGP session, assigning properties to the session, and defining packet and route filters. In practice, providers vary somewhat in their exact data requirements and configuration rules, particularly with regard to optional BGP commands. Throughout the discussion, we refer to the specific Cisco IOS commands necessary to provision the two customers. In contrast to vendor books and online documentation, we explain how to use the commands to solve specific problems and discuss how the commands relate to each other. In practice, the provider's provisioning system could generate router configurations in a vendor-neutral format such as the Routing Policy Specification Language (RPSL) [10] and then generate the vendor-specific representation using a tool such as RtConfig [11]. However, for the sake of simplicity, we focus on generating the raw configuration commands in a single step.

3.1 Customer-Specific Data

Provisioning a new BGP session requires the customer to answer basic technical questions, as highlighted in Table 1. The table shows example values for two customers corresponding to the configlets in Figure 3 and Figure 4, respectively; the first customer ("AT&T Research") has a single access link, whereas the second customer ("MoneyBags") has two access links. The basic information about the customer consists of a textual description, an AS number, and a list of IP address blocks. In some cases, the customer may have its own AS number (e.g., AS 6431 for AT&T Research); this is common for customers that connect to multiple upstream providers. In other cases, the AS number for the customer's BGP sessions is selected by the provider. The provider might have a dedicated AS number for BGP-speaking customers that do not have their own AS number [12] or use a private AS number in the range 64512–65535 [7]. For example, the provider could select a private AS number of 65000 for MoneyBags. In some cases, the customer may have its own blocks of IP addresses. Otherwise, the provider may function as a Local Internet Registry that assigns address blocks to customers out of its own larger block; this process can be automated based on address information the provider's provisioning database. As part of the address assignment process, the provider may add the customer's address assignment information to the appropriate "whois" database managed by the Regional Internet Registry.

The customer indicates its geographic location (e.g., Florham Park, NJ) and the desired media type and speed of each of the access links (e.g., 45 Mbps serial link). This information is used to select a router at a nearby homing location (e.g., Newark, NJ) that can accommodate the links, as highlighted in Table 2. The decision may depend on both the geographic proximity and the available slots on the production routers. The selection of the homing locations may, in turn, affect the assignment of IP address blocks to the customer to maximize route aggregation. Once the slot locations are selected, each interface is assigned an IP address and mask. In practice, a provider's interfaces may be numbered out of a special address block owned by the AS, with care to ensure that no two interfaces have the same IP address. Each interface is also assigned an Access Control List (ACL) identifier for the packet filter that determines which IP packets are permitted across the link. The assignment of the interface IP addresses and ACL identifiers can be automated based on inventory information stored in the provider's provisioning database. For example, the system could automatically select

	Variable	Example #1	Example #2	
Per-customer	Customer description	AT&T Research	MoneyBags	
	Customer AS number	6431	—	
	Customer prefixes	135.205.0.0/16	192.0.2.0/24	
Per-interface	Geographic location	Florham Park, NJ	LA, CA	LA, CA
	Media	serial link	POS	POS
	Speed	45 Mbps	155 Mbps	155 Mbps
Per-session	Number of intermediate device	0	0	
	Loopback, if multihop	—	12.107.34.21	
	Choice of inbound route-map	CUSTFACE	CUSTFACE	
	Choice of outbound route-map	ALLROUTES	ALLROUTES	
	Default route (yes/no)	no	no	
	Smaller BGP timers (yes/no)	no	yes	

Table 1: Customer technical questionnaire (TQ) for BGP provisioning

	Variable	Example #1	Example #2	
Per-interface	PR router name	ewr-nj	lax-ca	lax-ca
	Interface name	Serial10/0/1	POS11/0/1	POS12/0/1
	Interface address	12.119.155.157	12.119.155.161	12.119.155.165
	Interface mask length	30	30	30
	Packet filter id	1304	1273	1273
Per-session	Provider AS number	7018	7018	
	Route filter id	106	111	

Table 2: Customer-specific identifiers assigned by the provider

the next available ACL number on the PR router terminating the access link.

Each BGP session is associated with one or more of the access links between a provider router and a customer router. Different BGP sessions with the same customer may have different configuration options. The provider router is associated with an AS number belonging to the provider. A multi-hop configuration is used if the customer has two or more access links, or connects to the provider via an intermediate device; in these situations, the customer must provide a remote IP address for its end of the BGP session. The provider assigns an available ACL identifier for the route filter that determines which prefixes the customer can advertise over the BGP session. For the import and export policies, the provider may have a small menu of options that are configured on the routers in advance of customer provisioning. For example, some customers may want to receive a complete BGP routing table (e.g., ALLROUTES) whereas others may only want to receive routes to the provider's other customers or a simple default route to 0.0.0.0/0. The customer may be also be able to select certain optional features, such as different values for the keep-alive and hold-down timers. For simplicity, the provider might offer a few candidate settings of the timer values rather than allowing the customer to select arbitrary values.

```
interface Serial10/0/1
    description AT&T Research
    ip address 12.119.155.157 255.255.255.252
    ip access-group 1304 in
!
router bgp 7018
    neighbor 12.119.155.158 remote-as 6431
    neighbor 12.119.155.158 description AT&T Research
    neighbor 12.119.155.158 route-map CUSTFACE in
    neighbor 12.119.155.158 route-map ALLROUTES out
    neighbor 12.119.155.158 prefix-list 106 in
!
ip prefix-list 106 seq 5 permit 135.205.0.0/16
access-list 1304 permit 135.205.0.0 0.0.255.255
access-list 1304 permit 12.119.155.156 0.0.0.3
!
```

Figure 3: Example "configlet" for a BGP-speaking customer with a single access link

3.2 Access Interfaces

Configuring the access interfaces is the first step in provisioning a new customer. We assume that the routers have been physically deployed on the provider and customer networks, and the layer-two links have been connected. Physically, an interface resides on a card that connects to a slot in the router's switching fabric. The interface name identifies the location of the interface in the router, as well as the media type. Most routers support a variety of media, including Serial (e.g., T1 or T3), HSSI (High-Speed Serial Interface), Ethernet, FDDI (Fiber Distributed Data Interface), ATM (Asynchronous Transfer Mode), and POS (Packet-Over-SONET). Each media type has diverse configuration options for low-level details such as encapsulation, framing, error checking, and link capacity; in practice, network operators may have a set of default commands for each media type. In addition, the interface configuration may include the selection of certain link-scheduling and buffer management policies, such as assigning the buffer size, enabling Random Early Detection (RED), or assigning link-scheduling weights for class-based queuing. Since the layer-two commands and the resource-allocation policies do not influence the BGP configuration, we omit them from our example for the sake of simplicity.

In Figure 3, the `interface Serial10/0/1` command moves into the context of the Serial10/0/1 interface. In our example, the interface section has three indented commands applied in sequence:

- **Comment field**: The `description` is a comment field, set using the description from the TQ.
- **IP address/mask:** The `ip address` command sets the address and mask length for the interface, as assigned by the provider.
- **Packet filter identifier:** The `ip access-group` command associates a filter with the interface to discard certain "in" packets arriving on this interface. The packet filter is identified by an access-list number assigned by the provider.

```
interface POS11/0/1
    description MoneyBags
    ip address 12.119.155.161 255.255.255.252
    ip access-group 1273 in
!
interface POS12/0/1
    description MoneyBags
    ip address 12.119.155.165 255.255.255.252
    ip access-group 1273 in
!
router bgp 7018
    neighbor 12.107.34.21 remote-as 65000
    neighbor 12.107.34.21 description MoneyBags
    neighbor 12.107.34.21 update-source Loopback0
    neighbor 12.107.34.21 ebgp-multihop 2
    neighbor 12.107.34.21 route-map CUSTFACE in
    neighbor 12.107.34.21 route-map ALLROUTES out
    neighbor 12.107.34.21 prefix-list 111 in
    neighbor 12.107.34.21 timers 15 45
!
ip route 12.107.34.21 255.255.255.255 POS11/0/1
ip route 12.107.34.21 255.255.255.255 POS12/0/1
!
ip prefix-list 111 seq 5 permit 192.0.2.0/24
access-list 1273 permit 192.0.2.0 0.0.0.255
access-list 1273 permit 12.119.155.160 0.0.0.3
access-list 1273 permit 12.119.155.164 0.0.0.3
access-list 1273 permit 12.107.34.21 0.0.0.0
!
```

Figure 4: BGP configuration for a customer with two access links (POS11/0/1 and POS12/0/1)

The "!" line is a null command used to denote the end of the interface section of the configuration. Later, other commands are used to associate the interfaces with the BGP session and to define the packet filter.

3.3 BGP Connectivity

Entering the BGP section of the PR's configuration requires using the `router bgp` command, using the provider's AS number (e.g., 7018) as the argument. Then, the individual `neighbor` statements assign properties to the session. The IP address in the `neighbor` statements refers to the customer's end of the BGP session. In our example, the address is selected in one of two ways, as shown earlier in Figure 2. If the session has one access link and no intermediate devices, the `neighbor` address corresponds to the other end of the access link, as illustrated in Figure 2(a) and the example in Figure 3. Configuring the session in this simple manner has the advantage that the provider can easily "rehome" the access link to a different PR router without requiring the customer to reconfigure its end of the session. In the example in Figure 2(a), suppose that the PR has loopback address 12.1.17.8 and the access link is moved to a different provider router with loopback 12.89.2.45. The

CR could continue to refer to the IP address of the provider's end of the access link (12.119.155.157) as the remote end-point of the BGP session, since the access link could retain its network address as it moves to a new location. However, if the CR communicated with the PR's loopback address, rehoming the link would require changing the CR's configuration to communicate with 12.89.2.45 rather than 12.1.17.8.

A more complex configuration is required if the BGP session has multiple access links or traverses an intermediate device. In this case, the remote IP address is the loopback address provided by the customer, as illustrated in Figure 2(b) and the example in Figure 4. This second approach requires the configuration of a static route for each access interface associated with the session. This is achieved using the `ip route` command. The arguments are the customer's remote IP address (with a /32 mask) and the name of the interface (e.g., `ip route 12.107.34.21 255.255.255.255 POS11/0/1`). Similarly, the CR would be configured to have a BGP session with the provider's loopback address. As a result, the CR expects to receive packets from the PR's loopback address. Using the `update-source Loopback0` command ensures that the PR uses its loopback address (e.g., 12.0.1.7 in Figure 2(b)) as the source address of the packets associated with the BGP session. Communicating with the customer's loopback address requires the BGP messages to travel an extra "hop" to reach the remote BGP end-point. This requires over-riding the default TTL value for the IP packets. For example, Figure 4 uses the `ebgp-multihop 2` command to assign an initial TTL value of 2. If the customer has one or more intermediate devices between the PR and the CR, a higher TTL value would be necessary; in this case, the argument to the `ebgp-multihop` command would come from the customer TQ.

3.4 BGP Session Properties

The rest of the `neighbor` commands assign basic properties to the BGP session. Two of the items relate to the identity of the customer:

- **Customer AS number:** The `remote-as` command assigns the remote end-point's AS number, as indicated in the TQ or assigned by the provider.
- **Comment field**: The `description` is a comment field, set using the description from the TQ.

The remainder of the `neighbor` commands associate specific policies with the BGP session:

- **Import/export policies:** Two `route-map` commands assign the import ("in") and export ("out") policies for the session. Although these policies may be quite complicated in practice, they are typically installed on the router in advance of provisioning new customers. The TQ allows the customer to select the import and export policies from a menu of pre-defined options.
- **Route filter identifier:** The `prefix-list` command assigns a filter that discards certain "in" routes arriving via this BGP session. The route filter is identified by an identifier assigned by the provider.

- **Dynamic default route:** The `default-originate` command sends the customer a dynamic default route (to 0.0.0.0/0) when the session is established. In our example, we assume that this command is used when requested by the client in the TQ.

- **Keep-alive/hold timers:** The `timers` command over-rides the default values of the keep-alive and hold timers (60 and 180 seconds, respectively). Smaller values result in faster detection of legitimate failures, at the risk of mistakenly terminating the session during periods of transient delay in delivering the keep-alive messages. Faster failure detection has greater utility when a customer has multiple BGP sessions at different PRs, or sessions with multiple upstream providers, since traffic can be diverted to alternate links when a session fails. In our example, we assume that smaller values (e.g., 15 and 45 seconds, respectively, as shown in Figure 4) are assigned when requested by the customer in the TQ.

3.5 Packet Filters

The packet filter on the access interfaces protects the provider's network, and the rest of the Internet, from erroneous data packets sent by the customer. The packet filter discards packets entering or leaving the interface based on the certain header fields, as defined in the associated `access-list` commands (e.g., `access-list 1304` in Figure 3). The arguments of each `access-list` command for the packet filter are the prefix and mask in dotted-decimal notation; in binary form, the mask has a 1 for each wild-card bit (e.g., a mask length of 30 would be represented as 0.0.0.3). In our example, the packet filter consists of a small set of permissible source IP addresses:

- **Customer prefixes:** The filter allows packets with a source address owned by the customer (e.g., 135.205.0.0/16 in the first `access-list 1304` command in Figure 3). Filtering incoming packets based on the customer's address blocks IP address can prevent denial-of-service attacks with "spoofed" IP addresses [13].

- **Access links:** The filter also permits packets with a source address associated with *any* of the customer's access links across all of its homing locations (e.g., 12.119.155.161/30 and 12.119.155.169/30 in Figure 4). This allows the customer to use tools such as "ping" to verify that the provider's end of an access link is reachable.

- **CR's end of the BGP session:** The filter permits packets with the source address of the customer's end of the BGP session. This ensures that the IP packets containing BGP messages from the customer are not discarded. In some cases, this IP address is already "covered" by the customer's prefixes or access links. Otherwise, a separate ACL command is necessary (e.g., the last `access-list` command in Figure 4).

Packets with source addresses outside of these three address blocks would be discarded. In practice, the provisioning system may try to reduce the number of `access-list` commands to lower the packet-handling overhead on the interface cards. This may involve removing ACL clauses that are subsumed by other clauses

or combining adjacent address blocks. For example, in `access-list` 1273 in Figure 4, the lines with 12.119.155.160/30 and 12.119.155.164/30 could be replaced by a single line

```
access-list 1273 permit 12.119.155.160 0.0.0.7
```

that permits all addresses in 12.119.155.160/29.

3.6 Route Filters

The route filter on the BGP session protects the provider's network, and the rest of the Internet, from erroneous routing advertisements sent by the customer. In contrast to the packet filter, the route filter refers to specific destination *prefixes* rather than individual source IP addresses. In reality, the inbound route filter is part of the import policy and, as such, could be specified in the inbound route-map. However, some aspects of route filtering are customer-specific. In order to maintain a small set of generic route-maps, it is useful to use a separate construct (the `prefix-list`) for the customer-specific filtering. In our example, the `prefix-list` command discards any routes that do not correspond to the customer's prefixes (e.g., 135.205.0.0/16 in Figure 3 or 192.0.2.0/24 in Figure 4); the syntax for the `ip prefix-list` command specifies the prefix and mask in dotted-decimal and integer notation, respectively. This route-filtering policy protects the network from misconfigurations on the customer's end of the BGP session. For example, a customer might mistakenly advertise a route to some other prefix learned from another provider; accepting this route might cause one provider to send traffic to the other via their common customer!

Our example route filtering policy is fairly restrictive since it does not permit the customer to advertise routes for any *subnets* of its address blocks. In practice, a route filtering policy that allows subnetting would be more complicated than this. A customer owning 135.205.0.0/16 might want to be able to advertise different BGP routes for 135.205.1.0/24 and 135.205.2.0/24 to achieve certain load balancing goals. Allowing these kinds of announcements would require the provider to use a looser filter that allows subnetting of the addresses. However, the provider may not want to allow the customer to advertise *all* possible subnets of the address blocks, since this could lead to an explosion in the size of the routing tables. Instead, the provider might have a policy that allows for limited subnetting of the customer's address blocks. For example, the command

```
ip prefix-list 106 seq 5 permit 135.205.0.0/16 ge 16 le 24
```

allows a range of mask lengths from 16 to 24. In practice, a provisioning system could have two categories of route filters—one basic configuration that specifies a simple list of the customer's address blocks for regular customers and another that allows certain subnets (e.g., all subnets up to a certain mask length) for certain customers that need greater flexibility. For ease of presentation, we focus on a simple policy that does not allow the customers to advertise subnets of their address blocks.

4 Database-Driven Provisioning System

The provisioning process has three main steps—collecting the customer-specific information, storing the data in a database, and using the data to generate a configlet—as illustrated earlier in Figure 1. In this section, we focus specifically on the database issues in the provisioning system. First, we describe how to represent the customer-specific data and how to construct a virtual view (explained later) of the customer configuration after applying the provisioning rules. Then we describe how to use this virtual view to generate the sequence of commands that should be applied to the provider's router.

4.1 Provisioning Database

The heart of the BGP provisioning system is a database that stores all of the customer-specific data—the information from the technical questionnaire (TQ) as well as the provider-assigned identifiers. In practice, this database might store a wide variety of data about the provider's network and its customers and support numerous applications such as billing, reporting, maintenance, and inventory management. For simplicity, we describe just the portions of the database that relate to provisioning new BGP-speaking customers. We assume that the data related to the provider's inventory and layer-2 configuration have already been stored in the database. We also assume that the route-map definitions have been stored in the database and installed on the production routers as part of the initial deployment of the equipment. Table 3 shows the proposed data schema using a relational database as an example. The primary key fields are marked by ^{P} and the foreign keys are marked by ^{F}. The general design guidelines can be applied on other type of databases.

4.1.1 Design Criteria

Deciding what data to store is the first critical design decision. On the one hand, the database could store the individual configuration commands applied to the production routers. However, this approach does not allow the system to adapt to changes in the configuration rules and ties the solution very tightly with the details of specific vendor commands. Instead, we store only the customer-specific information (shown in **bold** font in Figure 3) and a small number of identifiers needed to link the data together. This has several advantages. First, fewer data elements need to be stored in the database. Second, data integrity is easier to maintain since there are minimal dependencies between the data elements. Third, the database schema is flexible enough to allow changes to the configuration rules and to support other network management applications, such as inventory management. The drawback of this approach is that the data elements that are not stored in the database must be generated every time they are needed. The generation of such data elements may involve expensive operations such as table joining.

The second major decision concerns how to structure the data in the database. One simple approach would be to create a single table where each field corresponds to a data element. However, this approach is unwieldy from a database development point of view. For example, a customer may have multiple address blocks that

Table	Fields	Data type	Examples				
Customer	Customer ID^{P}	Number	1	2			
	AS number	Number	6431	65000			
	Description	String	AT&T Research	MoneyBags			
	Geographic location	String	Florham Park, NJ	LA, CA			
Inventory	**Router**^{P}	String	ewr-nj	sfo-ca			
	Type	String	Cisco	Cisco			
	Location	String	Newark, NJ	LA, CA			
Device	Router^{P,F}	String	ewr-nj	lax-ca	lax-ca		
	Interface^{P}	String	Serial10/0/1	POS11/0/1	POS12/0/1		
	Slot	Number	10	11	12		
	Media	String	Serial	POS	POS		
	Speed	Number	45	155	155		
Link	Router^{P,F}	String	ewr-nj	lax-ca	lax-ca		
	Interface^{P,F}	String	Serial10/0/1	POS11/0/1	POS12/0/1		
	Prefix ID^{F}	Number	3	4	5		
	Packet filter ID	Number	1304	1273	1273		
BGP session	Session ID^{P}	Number	1	2			
	Route filter ID	String	106	111			
	Loopback IP address	Number	Nil	12.107.34.21			
	Inbound route-map	String	CUSTFACE	CUSTFACE			
	Outbound route-map	String	ALLROUTES	ALLROUTES			
	# of intermediate devices	Number	0	0			
	Smaller timers	Boolean	no	yes			
	Default originate	Boolean	No	No			
Assignment	Customer ID^{P,F}	Number	1	2	2		
	Session ID^{P,F}	Number	1	2	2		
	Router^{P,F}	String	ewr-nj	lax-ca	lax-ca		
	Interface^{P,F}	String	Serial10/0/1	POS11/0/1	POS12/0/1		
IP prefixes	Prefix ID^{P}	Number	1	2	3	4	5
	Type	String	Customer	Customer	BGP link	BGP link	BGP link
	Prefix	Number	135.205.0.0	192.0.2.0	12.119.155.157	12.119.155.161	12.119.155.165
	Netmask	Number	16	24	30	30	30
Prefix allocation	Prefix ID^{P,F}	Number	1	2	3	4	5
	Customer ID^{P,F}	Number	1	2	1	2	2

Table 3: The BGP provisioning data schema (the fields that store the customer-specific information are shown in **bold** font; the primary key fields are marked by ^{P} and the foreign keys are marked by ^{F}).

need to be included in the packet filter associated with access interfaces. However, each data record in the table can have a single value for each field, implying the need for multiple records to store the data for this customer. This involves repeating all of the other information for this customer across multiple rows in the table. This requires more disk space and makes it difficult to maintain the integrity of the data. Instead, we normalize the tables by breaking the data elements into several tables, where each of the eight tables in Table 3 corresponds to a important element of BGP configuration. This semantics-driven schema provides a meaningful grouping of the data elements and eases data interpretation. Since semantically-related data elements are likely to be needed at the same time, our schema should reduce the cost of querying the database.

4.1.2 Data Schema

In the design of the schema in Table 3, we keep efficiency in mind while avoiding assumptions about the details of the configuration rules. The schema has the following tables:

Customer: The *Customer* table stores the basic customer information—*Description*, *AS number*, and *Geographic location*—from the TQ. Ideally, one of these items (or some subset of the items) could serve as the primary key for this table. However, these fields are not necessarily unique. A single customer may have multiple AS numbers and the provider might use a single, dedicated AS number for multiple customers. A customer might have multiple locations, each corresponding to a different site. The description field is a string that is not necessarily unique across different customers. It may be possible to construct a primary key using all three of these data elements. However, this would be an inefficient solution and would result in multiple different primary key fields for a customer that has multiple AS numbers or multiple locations. Instead, we introduce a new field *Customer ID* that is generated when the entry is created. This field serves as the primary key for the *Customer* table. Although storing an extra field consumes additional storage space, this approach greatly improves the efficiency of the database operation.

Inventory and Device: Information about the equipment in the provider's network could be stored in a single table. However, a router usually has multiple interfaces, making it appealing to have one table for routers (*Inventory*) and another for interfaces (*Device*). This avoids duplicating the router-specific information (such as the router type and location) across all of the interfaces. The router name *Router* uniquely identifies each data record in the *Inventory* table and, as such, serves as a primary key. The *Device* table includes the router name along with the interface name, slot number, media type and link speed. The interface name and slot number have only local significance in the context of the specific router. As such, the primary key of the *Device* table is the combination of the router name and the interface name. The *Router* field also serves as a foreign key pointing to the *Router* field in the *Inventory* table. The foreign key enforces data integrity between the records in *Device* and *Inventory* tables—that is, each record stored in the *Device* table must correspond to a router stored in *Inventory* table. In addition, the foreign key facilitates join operations between the two tables to extract information such as the geographic location of a particular interface.

Link: In contrast to the physical information in the *Device* table, the *Link* table stores logical information about the configuration of access interfaces. As with the *Device* table, the primary key for the *Link* table is the combination of the router name and the interface name. The *Prefix ID* field is a foreign key to the *Prefix ID* in the *IP prefixes* table that indicates the address and mask assigned by the provider. We represent this information separately since the address prefix may be used in other contexts, such as an ACL specification. The *Packet filter ID* stores the packet filter identifier assigned by the provider. Assigning this identifier involves checking the database to select an available ACL number on the associated router. On the surface, the logical and physical information could be stored in a single table. However, the configuration of a link may change over time and different kinds of links may have different attributes. For example, a provider network has a large number of backbone links that participate in the intradomain routing protocol (such as OSPF or IS-IS); the configuration of a backbone link would include the OSPF/IS-IS weights and area but would not necessarily include a packet filter.

BGP session: The table *BGP session* stores all of the information related to the customer BGP sessions including the customer's loopback IP address (if known), route filter ID, inbound and outbound route-maps, number of intermediate devices between the two BGP speakers, the selection of smaller BGP timers, and the selection of the default-originate option. From the provider's viewpoint, the BGP session is uniquely identifiable from the PR name and the IP address of the remote end of the BGP session. However, the remote IP address is selected based on the configuration rules (e.g., it could correspond to the customer's end of an access link or the CR's loopback address). As such, the router name and remote IP address do not form a good primary key for the BGP session. Instead, we define an additional *Session ID* field that serves as the primary key for identifying records in the *BGP session* table.

Assignment: The association between the customers, links, and sessions is stored in the *Assignment* table. Having a separate assignment table provides the most flexibility in associating links and sessions with customers. Typically, a link is associated with at most one BGP session but this is not necessarily true. The PR may have BGP sessions to multiple CRs that connect (through an intermediate device) via a single access link. Although our example provisioning rules do not allow for this case, the database design should not preclude this kind of configuration. Similarly, a single access link could conceivably connect to multiple customers (say, over a shared medium such as an Ethernet or an FDDI ring), so we do not assume that a *Link* is associated with a single customer. The *Assignment* table allows arbitrary associations between customers, BGP sessions, and access links, and the table has three primary key fields—*Customer ID*, *Session ID*, and *(Router, Interface)*. These fields also serve as foreign keys to the *Customer*, *Session*, and *Link* tables, respectively.

IP prefixes and prefix allocation: IP addresses and IP prefixes arise in numerous contexts in the customer configuration—the numbering of the interfaces, the specification of packet and route filters, and the identification of the remote end-point of the BGP session. Often, the same prefix is instantiated in multiple places in the configlet. For example, the customer address blocks appear in the packet filter and the route filter. To avoid duplication, we store the prefixes in a separate table (*IP prefix*) and assign each prefix a unique identifier (*Prefix ID*), which also serves as the primary key for the table. The other fields identify the address and mask. We represent the mask as an integer mask length (e.g., 30) rather than choosing an inefficient and context-dependent dotted-decimal representation (e.g., 255.255.255.252 or 0.0.0.3). The *Type* field indicates whether the IP address block is allocated to a customer or to a particular kind of link. This is useful for other supporting applications that manage the allocation of the provider's address space. The allocation of address blocks to specific customers is maintained in the *Prefix allocation* table, which has the *Prefix ID* and the *Customer ID* as its primary key and its only fields. Having a separate table avoids including a *Customer ID* field in every entry of the *IP Prefix* table and also allows for the possibility in the future that an address block could be shared by two or more customers.

4.1.3 Virtual Views

Because the database stores only the essential data elements, other configuration data must be derived using the provisioning rules. This has the advantage that the system is flexible enough to support changes to these rules. However, some network management applications may need information about the configuration details. For example, an application might inspect the packet filter associated with each access interface to make sure that these filters stay within limitations imposed by certain kinds of routers. Ideally, the authors of these kinds of applications would not need to incorporate the details of the TQ or the provisioning rules, since these may be complicated and change over time. At one extreme, these applications could be written on top of a detailed log of the configuration commands, as dumped from each of the production routers. However, this approach requires parsing, modeling, and querying the raw configuration data, dealing with the syntactic differences in commands across different router products and versions, and recreating (where possible) the association of sessions and links with individual customers.

Instead, it is appealing to write these applications on top of a virtual view of how the customer is configured, without the raw details of the syntax of the commands. The configuration rules are enforced in the *creation* of views and the applications can be built *on top* of the views. Virtual views simplify the querying the database and isolate the applications from the physical tables. The virtual views can be organized around the main constructs in BGP customer configuration. Table 4 shows an example view organized around BGP-speaking customers. The view is constructed by joining physical tables using the foreign key mappings. The view shows not only the data elements stored in the database, but also the configuration data derived based on the configuration rules. We show the derived data in *italics*. For example, the view *Customer data* shows all the configuration data related to a customer, including the *Session IP address* and the *update-source* commands which are derived using the provisioning rules. It can be generated using a query similar to the one shown in Figure 5. The higher-level applications can query the database through the virtual views rather than accessing the physical tables. This hides the details of the configuration rules from the applications and make the queries much simpler. For example, the SQL query

```
SELECT *
FROM Customer_data
WHERE Customer_AS_number = '6431';
```

extracts all of the access interfaces and BGP sessions associated with the customer with AS 6431. It uses the virtual view to select the customer data from the (virtual) table shown in Table 4. The equivalent query that accesses several physical tables to extract the same information is shown in Figure 5. This query requires significant data manipulation and a detailed understanding of the configuration rules.

4.2 Configlet Generation

Configlet generation is one of the many applications that can be built on top of the virtual views. Before generating a configlet, the provisioning system checks the completeness and consistency of the data. Then, the

Name	Table	Fields	Example
AS Number	Customer	AS number	6431
Description	Customer	Description	AT&T Research
Geographic location	Customer	Geographic location	Florham Park, NJ
Router	Inventory	Router	ewr-nj
Location	Inventory	Location	Newark, NJ
Interface	Link	Interface	Serial10/0/1
Interface IP address	IP prefixes	Prefix + 1	12.119.155.157
Packet filter ID	Link	Packet filter ID	1304
Session IP address	BGP session	Loopback IP address (if !Nil)	12.119.155.158
	IP prefixes	Prefix + 2 (else)	
Route filter ID	BGP session	Route filter ID	106
Inbound route-map	BGP session	Inbound route-map	CUSTFACE
Outbound route-map	BGP session	Outbound route-map	ALLROUTES
EBGP multihop	BGP session	# of intermediate devices (if > 1)	—
Update source	BGP session	Loopback0 (if EBGP multihop > 1)	No
Default originate	BGP session	Default originate	No
BGP keep-alive timer	BGP session	BGP keep-alive timer	—
BGP hold timer	BGP session	BGP hold timer	—
Prefix	IP prefixes	Prefix	135.205.0.0
Netmask	IP prefixes	Netmask	16

Table 4: The example view *Customer data* created in BGP provisioning database.

system uses the data from the virtual view to populate a template of configuration commands.

4.2.1 Data Completeness and Consistency

The customer-specific data are propagated into the database in several steps that take place at different times and may involve different applications. The provider's inventory data are stored in the database before provisioning new customers. Data from the TQ enter the database after the customer provides the information. Other customer-specific data, such as ACL identifiers and interface IP addresses, are generated by the provider and stored back in the database. In addition, information in the database may be modified or removed by other network management applications. Before generating the configlet for a customer, the provisioning system needs to check that all the necessary data have been stored in the database. Otherwise, the provisioning system might generate an incomplete or incorrect configlet for the customer. Checking the completeness and consistency of the data involves inspecting the virtual view for each of the configuration components (e.g., interface and session) and ensuring that all the fields are populated. Underneath the virtual view, this check follows the various foreign keys to ensure that all of the data elements are linked together.

4.2.2 Configuration Template

The virtual view already incorporates the process of applying the configuration rules to the customer-specific data. However, the view does not include the details of the *syntax* of the commands use to configure the production router. A template can be used to automatically generate the raw configuration commands. The

```
SELECT Customer.AS_number, Customer.Description, Customer.Geographic_location,
  Inventory.Router, Inventory.Location, Link.Interface,
  SUBSTR(IP_prefixes.prefix, 1,
    INSTR(IP_prefixes.prefix, '.', 1, 3) -1) || '.'  ||
    TO_CHAR(TO_NUMBER(SUBSTR(IP_prefixes.prefix, '.', 1, 3) + 1)) + 1),
  Link.Packet_filter_ID,
  DECODE(BGP_session.Loopback_IP_address, ' ',
    SUBSTR(IP_prefixes.prefix, 1,
    INSTR(IP_prefixes.prefix, '.', 1, 3) -1) || '.'  ||
    TO_CHAR(TO_NUMBER(SUBSTR(IP_prefixes.prefix, '.', 1, 3) + 1)) + 2),
    BGP_session.Loopback_IP_address),
  BGP_session.Route_filter_ID,
  BGP_session.Inbound_route_map, BGP_session.outbound_route_map,
  BGP_session.#_of_intermediate_devices,
  DECODE(BGP_session.#_of_intermediate_devices, '0', ' ', 'Loopback0'),
  BGP_session.Default_originate, BGP_session.BGP_keep_alive_timer,
  BGP_session.BGP_holder_timer
FROM BGP_session, IP_prefixes, Customer, Inventory, Link, Assignment
WHERE Customer.AS_number = '6431'
  AND Customer.Customer_ID = Assignment.Customer_ID
  AND Assignment.Session_ID = BGP_session.Session_ID
  AND Assignment.Router = Inventory.Router
  AND Assignment.Router = Link.Router
  AND Assignment.Interface = Link.Interface
  AND Link.Prefix_ID = IP_Prefixes.Prefix_ID
```

Figure 5: SQL query to extract all the interfaces and BGP sessions assigned to customer AS 6431.

template is dependent on the router product, and in some cases even the version of the router. Figure 6 presents a Cisco IOS template that can be used for our example configuration rules. In the template, the data elements provided by the provisioning system are specified as variables denoted by "<...>", where the name of the variable corresponds to a field in the virtual view for that section of the configuration. The optional commands are bracketed with "[" and "]" characters; the use of these commands depends on whether the variable has a value. For example, the `update-source` command is not used when the field has a Null value (denoted by "—" in the last column of Table 4); otherwise, the command is used and the "<update-source>" variable is set to the value of this field (e.g., `Loopback0`). Each section ends with a line with a "!" character.

The process of populating the configuration template can be automated. A parser can read the template section by section. For each section, the parser pulls the corresponding data elements from the provisioning database through the virtual views. The parser goes through each command in the section, computes the value of the variables based on the data elements, assigns the value to the variables, and prints the command. In some cases, a variable such as <interface> may appear multiple times for the same customer. When this occurs, the parser iterates through the set of commands in a section multiple times. For example, if a customer has two interfaces associated with one BGP session, the parser goes through the interface section twice and prints two set of interface commands, one for each interface. The parser prints the optional commands (e.g.,

```
hostname <router>
!
interface <interface>
    description <cust-description>
    ip address <interface-ip> <link-mask>
    ip access-group <packet-filter-acl> in
!
router bgp <provider-as-number>
    neighbor <session-ip> remote-as <customer-as-number>
    [neighbor <session-ip> description <description>]
    neighbor <session-ip> route-map <inbound-routemap> in
    neighbor <session-ip> route-map <outbound-routemap> out
    neighbor <session-ip> prefix-list <route-filter-acl> in
    [neighbor <session-ip> update-source <update-source>]
    [neighbor <session-ip> ebgp-multihop <ebgp-multihop>]
    [neighbor <session-ip> default-originate]
    [neighbor <session-ip> timers <keep-alive> <hold>]
!
access-list <packet-filter-acl> permit <cust-prefix> <cust-mask-wild>
access-list <packet-filter-acl> permit <link-prefix> <link-mask-wild>
[access-list <packet-filter-acl> permit <session-ip>]
ip prefix-list <route-filter-acl> seq <seq-num> permit <cust-prefix>/<cust-mask-int>
permit ip host <cust-prefix> host <cust-mask>
[ip route <session-ip> 255.255.255.255 <interface>]
!
```

Figure 6: Cisco IOS configuration template for a BGP session

`update-source`) only if the corresponding data elements are present in the database.

Some data elements, such as the masks on the link and customer address blocks, may appear in multiple formats in the configlet. In such a case, the data elements are converted into the correct format depending on the command the parser is processing. For example, the link mask /30 appears in the command `ip address <interface-ip> <link-mask>` in the interface configuration and in the command `access-list <packet-filter-acl> permit <link-prefix> <link-mask-wild>` in the packet filter definition. After extracting the value from the database, the parser converts the mask length of "30" to 255.255.255.252 for the `<link-mask>` (in the `interface` section) and to 0.0.0.3 for the `<link-mask-wild>` (for the packet-filter definition). The name of the corresponding parameter in the ACL commands indicates whether or not the mask will be converted to 255.255.255.252 or 0.0.0.3. A `wild` in the parameter's name indicates that the mask will be converted to 0.0.0.3, whereas an `int` in the parameter's name (e.g., `<cust-mask-int>` in the `ip prefix-list` command for the route filter) indicates that the mask will be presented as an integer. Thus, the configuration commands are automatically generated by populating the data elements into the configuration template.

5 Conclusion

Manual configuration of IP routers is an expensive, time-consuming, and error-prone process. In this paper, we have described an approach to automated provisioning of BGP-speaking customers with a three-step process—gathering customer-specific data, storing the information in a database, and generating configuration commands by applying provisioning rules. We presented a case study example of a Technical Questionnaire (TQ) for new customers, a database schema for storing the customer-specific information, configuration rules for driving the provisioning process, and a template capturing the syntax of the configuration commands. In practice, providers may vary in the details of their TQ, database technologies, provisioning rules, and router command sets but the basic approach remains the same. Based on the work in this paper, we have developed a prototype implementation of our provisioning system, using configuration data from AT&T's commercial IP backbone to bootstrap the database.

A complete solution to the provisioning problem would need to address several other issues. First, the system should automatically generate the small set of route-maps on the production routers; in practice, the route-maps can be quite complicated and should be compiled from a higher-level specification of the intended policies. Second, the system should allow the configuration to proceed in stages, with physical testing at each step; each stage could involve applying a subset of our configlet and testing the resulting set-up. Third, the system should support changes such as removing a customer, migrating a customer from one router to another, adding one or more new links, or changing the routing policy; a complete solution would require configuration rules for each of these "use cases." Fourth, the system should support a smooth transition from manual to automated configuration by loading existing customers into the system; this would involve extracting the relevant data from the router configuration files, after a "clean-up" phase to ensure that the actual PR configuration adheres to the rules. Fifth, although our system reduces dependency on the router vendor products by representing the command syntax in a separate template, a complete solution should have a vendor-neutral layer and separate "drivers" to generate the commands for each router product and version. Sixth, for ease of extensibility, the provisioning rules should be expressed separately from the database in a language with constructs that appeal to network operators. In our future work, we plan to use our provisioning prototype and a detailed analysis of AT&T's configuration data to explore many of these open issues.

Acknowledgments

We would like to thank Jay Borkenhagen and Rich Kwapniewski for their generous help in understanding Cisco router configuration and the BGP routing policies for a large service provider. Thanks also to Nick Feamster, Tim Griffin, Flip Korn, Rich Kwapniewski, Zhuoqing Morley Mao, Aman Shaikh, and the anonymous reviewers for their comments on the paper.

References

[1] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1771, IETF, March 1995.

[2] J. W. Stewart, *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley, January 1999.

[3] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfigurations," in *Proc. ACM SIGCOMM*, August 2002.

[4] S. Halabi and D. McPherson, *Internet Routing Architectures*. Cisco Press, second ed., 2001.

[5] I. van Beijnum, *BGP: Building Reliable Networks with the Border Gateway Protocol*. O'Reilly, September 2002.

[6] IANA, "Special-Use IPv4 Addresses," RFC 3330, IETF, September 2002.

[7] J. Hawkinson and T. Bates, "Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)," RFC 1930, IETF, March 1996.

[8] G. Huston, "Interconnection, peering, and settlements," in *Proc. INET*, June 1999.

[9] E. Chen and J. Stewart, "A Framework for Inter-Domain Route Aggregation," RFC 2519, IETF, February 1999.

[10] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra, "Routing Policy Specification Language (RPSL)," RFC 2622, IETF, August 1999.

[11] D. Meyer, J. Schmitz, C. Orange, M. Prior, and C. Alaettinoglu, "Using RPSL in Practice," RFC 2650, IETF, June 1999.

[12] J. Stewart, T. Bates, R. Chandra, and E. Chen, "Using a Dedicated AS for Sites Homed to a Single Provider," RFC 2270, IETF, January 1998.

[13] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing," RFC 2267, IETF, January 1998.