1

The Border Gateway Protocol

When networks were small, there was no concept of interior and exterior gateway protocols; a network ran a routing protocol, and that was the end of it. The Internet, for instance, ran the *Hello Protocol* on devices called fuzzballs (before they were called routers), until some problems in the Hello Protocol led to the development of RIP (Routing Information Protocol). RIP was run as the only routing protocol on the Internet for many years. Over time, however, the Internet grew (and grew and grew), and it became apparent that something more was needed in routing protocols–a single ubiquitous protocol couldn't do all the work that routing protocols were being required to do and scale in any reasonable manner.

In January 1989 at the 12th IETF meeting in Austin, Texas, Yakov Rekhter and Kirk Lougheed sat down at a table and in a short time a new exterior gateway routing protocol was born, the Border Gateway Protocol (BGP). The initial BGP design was recorded on a napkin rumored to have been heavily spattered with ketchup. The design on the napkin was expanded to three hand-written sheets of paper from which the first interoperable BGP implementation was quickly developed. A photocopy of these three sheets of paper (See Appendix B) now hangs on the wall of a routing protocol development area at Cisco System in Santa Clara, CA.

From this napkin came the basis for BGP as we know it today. Now, with countless contributors and hundreds of pages in tens of documents, deployed in thousands of networks, interdomain routing in the Internet today is defined as BGP.

This book is about BGP, from the basics of the BGP protocol itself to information on deploying BGP in networks stretching from small and simple to very large and extremely complex. We'll begin with an overview of the BGP protocol itself here in Chapter 1. We'll then move into various deployment situations, starting with small enterprise networks using BGP internally and to connect to the Internet. From there we'll continue to move through ever-larger scale deployments of BGP, discussing how BGP and its extensive policy mechanisms fit into network architectures. We continue by providing details about finely tuning BGP to perform optimally and scale effectively in an array of deployment scenarios. We finish with in-depth discussions on debugging and troubleshooting various problems within the protocol and BGP networks.

Exterior and Interior Gateway Protocols

In order to understand why BGP is designed the way it is, you first need to understand where it fits in the world of routing protocols. Routing protocols can be divided along several axes, the first being Interior Gateway Protocols (IGPs) versus Exterior Gateway Protocols (EGPs). The primary difference between EGPs and IGPs is the place in the network where they provide reachability information; that is, within an administrative routing domain (intradomain) or between administrative routing domains (interdomain).

Routing Domains

Exactly what a routing domain is depends primarily on the context. In Intermediate System to Intermediate System (IS-IS) terminology, for instance, a *routing domain* is the area in which topology information is flooded. Open Shortest Path First (OSPF) simply refers to this as an *area.* Within the context of BGP, however, a routing domain is *the set of routers under the same administrative control.* In other words, there are routers your company, school, division, and so on can administer, configure, and manage, and there are routers beyond your control. Those routers under your control are typically said to be within your routing domain; those outside your control are outside your routing domain.

This definition isn't as precise as it sounds, since a particular router may be within the control of an entity, but not under the control of everyone who works for that entity or is a part of that entity. For example, a limited set of people within an organization may be able to configure the router that connects that organization to the Internet, but that doesn't necessarily mean this router is in a separate routing domain from the rest of the routers in the organization.

Within the world of BGP, those routers under a single point of administrative control are referred to as an *autonomous system (AS)*. Exterior routing, then, concerns itself with providing routing information between routing domains, or autonomous system boundaries while interior routing concerns itself with providing routing information within a routing domain or autonomous system.

Why Not Use a Single Protocol for Both Internal and External Routing?

If all routing protocols provide the same information–reachability and path information–why not use a single routing protocol for both interior and exterior routing? The simple answer is that routing protocols may not just provide reachability information–they may also provide policy information. There are several reasons why protocols designed to route within an autonomous system don't carry policy information:

- Within an autonomous system (AS), policy propagation generally isn't important. Since all the routers contained within the routing domain are under a single administrative control, policies can be implemented on all the routers administratively (through manual configuration). As such, the routing protocol doesn't need to propagate this information.
- Speed of convergence is a very important factor for routing protocols within an autonomous system, while it is not as much of a factor as stability between autonomous systems. Routing protocols providing reachability information within an autonomous system need to be focused on one thing: providing accurate information about the topology of the network as quickly and efficiently as possible. Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), and Enhanced Interior Gateway Protocol (EIGRP) all provide this sort of routing, expressly designed for intradomain routing.

Some policy propogation is creeping into interior gateway protocols in the form of information about the quality of service across various paths within a network; even here, the definitions of interior and exterior routing becomes blurred.

Why is it so important to split the routing information learned from within your domain from the routing information learned from outside your domain? There are many reasons–for instance, in order to scope propagation of changes made within a routing domain so they don't impact external routing domains, or perhaps to provide the capability to hide specific information about your network from external entities. The reasoning behind these and many other possible responses will become more obvious as we proceed through the book.

Preventing Changes in Other Routing Domains from Impacting Network Operation

Let's examine the network illustrated in Figure 1.1 and consider how changes in one routing domain could have a serious negative impact on the operation of another routing domain.

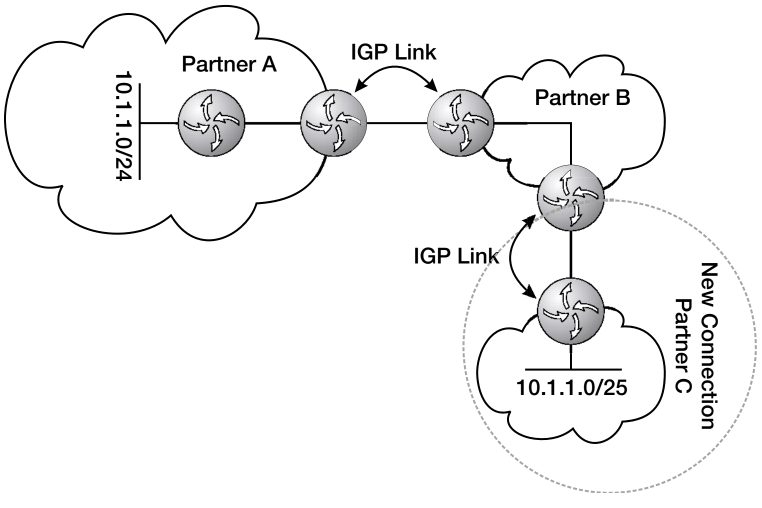

Figure 1.1
Unintentional consequences of bringing up a new link when sharing routing information.

In this network, the network administrators have decided to share routing information through an interior gateway protocol, including specific information about how to reach servers and hosts within each other's networks as needed. It's decided that 10.1.1.0/24 is one of the destinations that they need to share information about, so redistribution between the IGPs used in Partner A and Partner B's networks is set up to allow this information to leak between the two routing domains. In time, Partner B also partners with Partner C and again uses IGP redistribution to share information about reachable destinations between the two routing domains.

However, in this case, the routing information provided by Partner C into Partner B's routing domain, and thus leaked into Partner A's routing domain, overlaps (or conflicts) with the internal routing information in Partner A's routing domain. The result is that some destinations within Partner A's network will become unreachable to sources within Partner A's network–the actions of Partner B's network administrators have caused a fault in Partner A's network. This sort of problem is not only difficult to identify, it is also difficult to fix, since it will involve actions on the part of the network administrators from, possibly, all three routing domains.

Hiding Information about Your Network

The network illustrated in Figure 1.1 also uncovers another problem which can result when simple IGP redistribution is used to share information between autonomous systems; in this case, information about Partner C's internal network infrastructure is passed on to Partner A. If Partner A and Partner C are actually competitors, the information about Partner C's network could actually be used to compromise their competitive position. In general, it is always best to use policy-based rules to prevent information about your internal network from leaking beyond its intended bounds.

Policies between Domains

Examining the issues illustrated through Figure 1.1, it is apparent that some sort of policy implemented by Partner A, in the first case, and by Partner C, in the second case, would prevent the problems described. For instance, in the first case, a policy of not accepting routing information from outside the network that would interfere with internal routing information would resolve this problem, and all such future problems, without manually configuring a list of filters on a regular basis. In this example, simply filtering the routing information learned by Partner A from Partner B so that no prefixes with a prefix length longer than 24 bits be accepted would resolve this issue permanently if all the networks within Partner A's routing domain have a 24-bit length.

In the second case, if Partner C could somehow mark the routing information it is advertising to Partner B so that Partner B will not pass the information on to Partner A, this problem could also be resolved without resorting to manual lists maintained by Partner B. So two possible policies we would want to implement between routing domains would be to mark routes so they cannot (or should not) be advertised beyond the adjacent routing domain (Partner B) and to prevent leaking information that would provide a better route to internal networks than the internal routing information provides. What other sorts of policies would we want to implement through an Exterior Gateway Protocol (EGP)?

- Always take the closest exit point. If you want to allow traffic from other networks to traverse your network but you want to minimize the amount of bandwidth you need to provision in order to allow this, then you should be able to set up a policy of always taking the closest exit point out of your network, rather than the best path, toward the destination. This is typically referred to as *closest-exit* or *hot potato routing.*
- Take the closest exit point to the final customer. In some cases, in order to provide better service to customers who are reaching your network through another autonomous system, you want to be able to always choose the best, or shortest, path to the final destination rather than the shortest path out of your network. This is typically referred to as *best-exit routing,* though oddly it's sometimes also referred to as *cold potato routing.*
- Take the cheapest exit point. In some cases, you may have contracts requiring payment per a given amount of traffic sent on a particular link or set of links. If this is true, you may want to route traffic out of your autonomous system based on the cheapest exit point rather than the closest.
- Don't traverse certain networks. If you are running a network carrying secure or sensitive data, you might want to have some control over the physical forwarding path the traffic takes once it leaves your network. In reality, controlling the path your traffic takes is almost impossible, even with BGP, because IP

packets are routed hop by hop, and thus anyone you send the packets to can decide to send them someplace you don't want them to go.

- Avoid accepting redundant or unstable routing information from other networks. In order to scope resource consumption within your network, you may want to impose policies that discard redundant routing information or suppress unstable route advertisement.

In some cases, combining two or more of these different policies may be required. For instance, you may want to take the closest cheap exit point, from you network, and not traverse certain other networks. These policy definitions are rather high level; they state goals rather than the implementation of goals. One of the more confusing aspects of deploying BGP is turning such goals into actual implemented policies within and at the borders of your network.

Distance Vector, Link State, and Path Vector

Routing protocols are effectively distributed database systems. They propagate information about the topology of the network among the routers within the network. Each router in the network then uses this distributed database to determine the best loop free path through the network to reach any given destination. There are two fundamental ways to distribute the data through a network:

- By distributing vectors, each router in the network advertises the destinations it can reach, along with information that can be used to determine the best path to each reachable destination. A router can determine the best vector (path) by examining the destinations reachable through each adjacent router or neighbor, combined with additional information, such as the metric, which indicates the desirability of that path. There are two types of vector-based protocols: distance vector and path vector.
- By distributing the state of the links attached to the routers, each router floods (or advertises to all other routers in the network, whether directly adjacent or not) the state of each link to which it is attached. This information is used independently by each router within the routing domain to build a tree representing a

topology of the network (called a shortest path tree). Routing protocols that distribute the state of attached links are called link state algorithms.

Each of these data distribution methods is generally tied to a specific method of finding the best path to any given destination within the network. The following sections provide a quick overview (or review) of each of these types of routing protocols. Remember that a primary goal of routing protocol design is that routing protocols must be capable of determining loop free paths through the network. Generally, routing protocols assume that the best (or shortest) path through the network is also loop free.

Link State

Link state protocols, such as IS-IS and OSPF, rely on each router in the network to advertise the state of each of their links to every other router within the local routing domain. The result is a complete network topology map, called a shortest path tree, compiled by each router in the network. As a router receives an advertisement, it will store this information in a local database, typically referred to as the link state database, and pass the information on to each of its adjacent peers. This information is not processed or manipulated in any way before it is passed on to the router's adjacent peers. The link state information is *flooded* through the routing domain unchanged, just as the originating router advertises it.

As each router builds a complete database of the link state information as advertised by every other router within the network, it uses an algorithm, called the *shortest path first algorithm,* to build a tree with itself as the center of that tree. The shortest path to each reachable destination within the network is found by traversing the tree. The most common shortest path first algorithm is the Dijkstra algorithm.

Distance Vector

Routers running distance vector algorithms advertise the vector (path) and distance (metric) for each destination reachable within the network to adjacent (directly connected) peers. This information is placed in a local database as it is received, and some algorithm is used to determine which path is the best path to each reachable destination. Once the best path is determined, these best paths are advertised to each directly connected adjacent router.

Two common algorithms used for determining the best path are Bellman-Ford, which is used by the Routing Information Protocol (RIP and RIPv2), and the Diffusing Update Algorithm (DUAL), used by the Enhanced Interior Gateway Protocol (EIGRP).

Path Vector

A path vector protocol does not rely on the cost of reaching a given destination to determine whether each path available is loop free. Instead, path vector protocols rely on analysis of the path to reach the destination to learn if it is loop free. Figure 1.2 illustrates this concept.

A path vector protocol guarantees loop-free paths through the network by recording each hop the routing advertisement traverses through the network. In this case, router A advertises reachability to the 10.1.1.0/24 network to router B. When router B receives this information, it adds itself to the path and advertises it to router C. Router C adds itself to the path and advertises to router D that the 10.1.1.0/24 network is reachable in this direction.

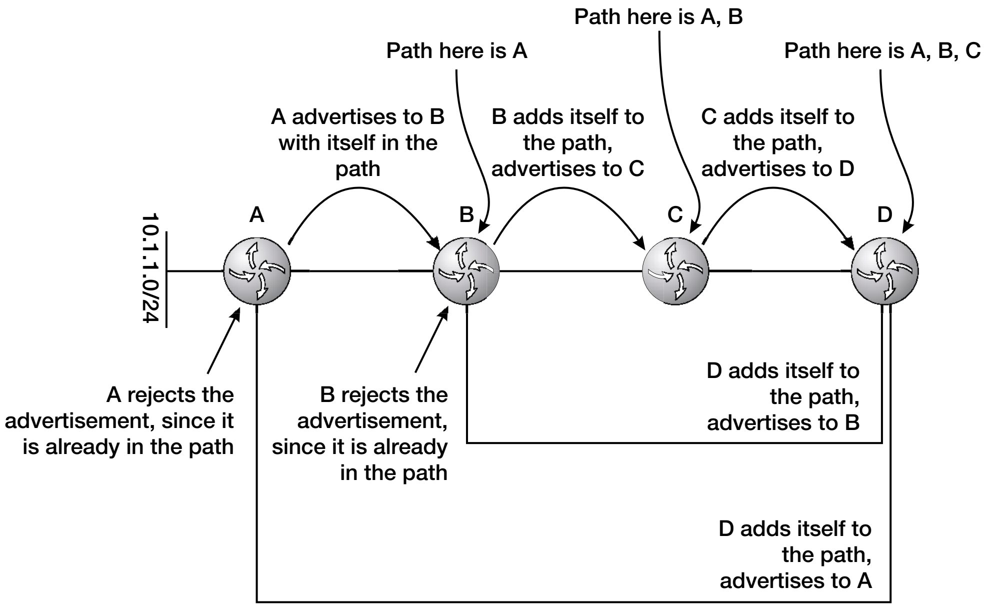
Path here is A, B
Path here is A
Path here is A, B, C
A advertises to B with itself in the path
B adds itself to the path, advertises to C
C adds itself to the path, advertises to D
A
B
C
D
10.1.1.0/24
A rejects the advertisement, since it is already in the path
B rejects the advertisement, since it is already in the path
D adds itself to the path, advertises to B
D adds itself to the path, advertises to A

Figure 1.2
Simple illustration of path vector protocol operation.

Router D receives the route advertisement and adds itself to the path as well. However, when router D attempts to advertise that it can reach 10.1.1.0/24 to router A, router A will reject the advertisement since the associated path vector contained in the advertisement indicates that router A is already in the path. When router D attempts to advertise reachability for 10.1.1.0/24 to router B, router B also rejects it since router B is also already in the path. Anytime a router receives an advertisement in which it is already part of the path, the advertisement is rejected since accepting the path would effectively result in a routing information loop.

BGP Path Vector Implementation

BGP implements the path vector concept on a larger scale rather than treating a single router as a single point in the path to any given destination. BGP treats each autonomous system as a single point on the path to any given destination (Figure 1.3).

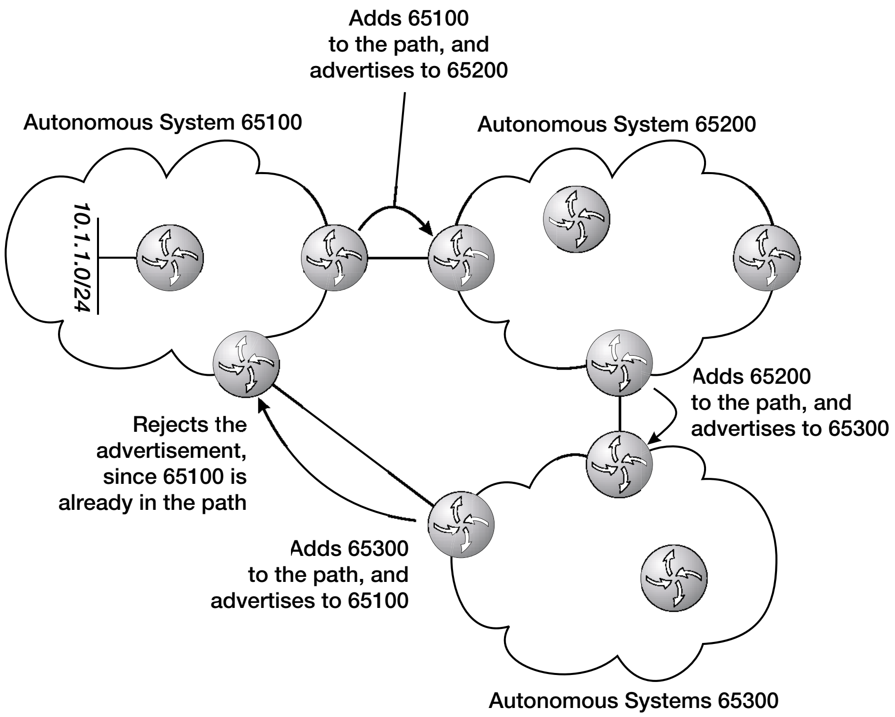
Adds 65100 to the path, and advertises to 65200
Autonomous System 65100
Autonomous System 65200
10.1.1.0/24
Adds 65200 to the path, and advertises to 65300
Rejects the advertisement, since 65100 is already in the path
Adds 65300 to the path, and advertises to 65100
Autonomous Systems 65300

Figure 1.3
Path vector over a set of autonomous systems.

This case is identical to the case in Figure 1.2, except that each autonomous system is considered a point along the path rather than a single router. The network 10.1.1.0/24, typically referred to as a *prefix,* is advertised with the list of autonomous systems the update has passed through; this list of autonomous systems is called the *AS Path.* AS 65100 originates the prefix 10.1.1.0/24, adding itself to the AS Path and advertises it to AS 65200. AS 65200 adds itself to the AS Path, and advertises the prefix to 65300. When AS 65300 advertises the prefix 10.1.1.0/24 to AS 65100, the prefix will be rejected since the 65100 sees that its local AS is already included in the AS Path, and accepting the route would result in a routing information loop.

The primary reason BGP treats an entire autonomous system as a single hop in the AS Path is to hide topological details of the AS. AS 65200, for instance, cannot tell what the path through AS 65100 looks like, only that the destination is reachable through AS 65100. One interesting side effect of treating each autonomous system as a single entity with which the autonomous system path vector is associated is that without additional information or rules, BGP can only detect loops between autonomous systems: it cannot guarantee loop-free paths inside an AS (Figure 1.4).

Since every router within AS 65200 receives the prefix 10.1.1.0/24 with the same AS Path, and BGP relies on the AS Path to prevent loops from forming, it is obvious that BGP cannot provide loop-free routing within an AS. As a result, BGP must ensure that every router in the AS

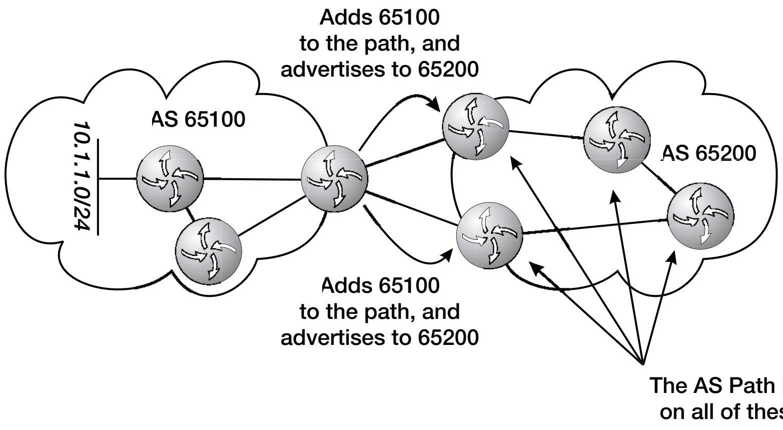

Figure 1.4
BGP routing within an AS.

makes the same decision as to which exit point to use when forwarding packets to a given destination and that a constrained set of route advertisement rules is used within the autonomous system. BGP then allows the interior gateway protocol running within the AS to determine the best path to each of the AS exit points.

BGP Peering

What are the mechanics of one BGP speaker peering with another speaker? What substrate protocols does BGP use to transport routing information? This section describes various aspects of BGP peering.

While BGP is most often run on routers, which are also responsible for forwarding traffic, in some cases other devices may run BGP as well, whether to simply gather information about the routing tables being carried in BGP or to carry routing information between routers. Since this is the case, we will sometimes refer to *devices that are running BGP* rather than *routers* specifically. A device that is running BGP is called a *BGP speaker,* and two BGP speakers that form a BGP connection for the purpose of exchanging routing information are called BGP *peers* or *neighbors.*

BGP Transport

How does BGP carry information about reachable destinations between the devices (routers) running BGP? How is the information encoded when it's transported between peers?

Transporting Data between Peers

A Transmission Control Protocol (TCP) transport connection is set up between a pair of BGP speakers at the beginning of the peering session and is maintained throughout the peering session. Using TCP to transport BGP information allows BGP to delegate error control, reliable transport, sequencing, retransmission, and peer aliveness issues to TCP itself and focus instead on properly processing the routing information exchanged with its peers.

When a BGP speaker first initializes, it uses a local ephemeral TCP port, or random port number greater than 1024, and attempts to contact each configured BGP speaker on TCP port 179 (the well-known

BGP port). The speaker initiating the session performs an active open, while the peer performs a passive open. It's possible for two speakers to attempt to connect to one another at the same time; this is known as a *connection collision.* When two speakers collide, each speaker compares the local *router ID* to the router ID of the colliding neighbor. The BGP speaker with the higher router ID value drops the session on which it is passive, and the BGP speaker with the lower router ID value drops the session on which it is active (i.e., only the session initiated by the BGP speaker with the larger router ID value is preserved).

BGP Routes and Formatting Data

A BGP route is defined as a unit of information that pairs a set of destinations with the attributes of a path to those destinations. The set of destinations is referred to~~,~~ by BGP~~,~~ as the Network Layer Reachability Information (NLRI) and is a set of systems whose IP addresses are represented by one IP prefix.

BGP uses *update messages* to advertise new routing information, withdraw previously advertised routes, or both. New routing information includes a set a BGP attributes and one or more prefixes with which those attributes are associated. While multiple routes with a common set of attributes can be advertised in a single BGP update message, new routes with different attributes must be advertised in separate update messages.

There are two mechanisms to withdraw routing information in BGP: To withdraw routes explicitly, one or more prefixes that are no longer reachable (unfeasible) are included in the withdrawn routes field of an update message (the update message may contain one or more new routes as well). No additional information, such as associated path attributes (e.g., AS Path), is necessary for the routes being withdrawn. Alternatively, because a BGP speaker only advertises a single best route for each reachable destination, a BGP update message that contains a prefix that has already been advertised by the peer, but with a new set of path attributes, serves an implicit withdraw for earlier advertisements of that prefix.

A BGP update message is made up of a series of *type-length vectors* (TLVs). Attributes carried within the BGP message provide information about one or more prefixes that follow; attributes are described in the BGP Attributes section later in this chapter.

BGP data, as it's transported between peers, is formatted as shown in Figure 1.5.

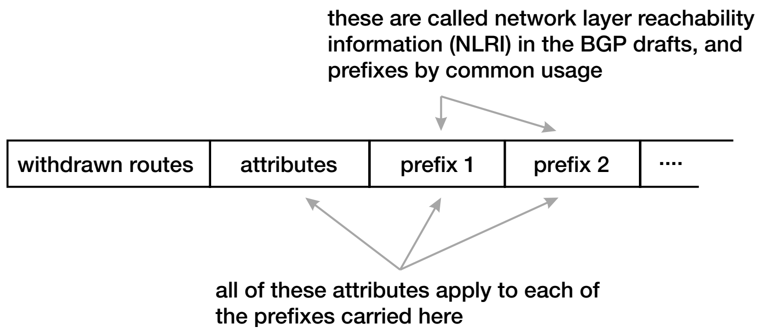
these are called network layer reachability information (NLRI) in the BGP drafts, and prefixes by common usage
withdrawn routes
attributes
prefix 1
prefix 2
....
all of these attributes apply to each of the prefixes carried here

Figure 1.5
Encoding information in a BGP packet.

As previously noted, one interesting aspect of this packet format is that while only a single set of attributes may be carried in each update message, many prefixes sharing that common set of attributes may be carried in a single update. This leads to the concept of *update packing,* which simply means placing two or more prefixes with the same attributes in a single BGP update message.

Interior and Exterior Peering

Beyond the mechanics of building peering relationships and transporting data between two BGP speakers, there are two types of peering relationships within BGP: *interior peering* and *exterior peering*. BGP sessions between peers within a single autonomous system are referred to as *interior BGP,* or iBGP, sessions, while BGP running between peers in different autonomous system are referred to as *exterior BGP,* or eBGP, sessions.

There are four primary differences between iBGP and eBGP peering relationships:

- Routes learned from an iBGP peer are not (normally) advertised to other iBGP peers. This prevents routing loops within the autonomous system, as discussed in the previous section titled BGP Path Vector Implementation.
- The attributes of paths learned from iBGP peers are not (normally) changed to impact the path selected to reach some outside network. The best path chosen throughout the autonomous system must be consistent to prevent routing loops within the network.

- The AS Path is not manipulated when advertising a route to an iBGP peer; the local AS is added to the AS Path only when advertising a route to an eBGP peer.
- The BGP next hop is normally not changed when advertising a route to an iBGP peer; it is always changed to the local peer termination IP address when a route is being advertised to an eBGP peer.

These last two points–the BGP next hop is normally changed when advertising a route to an eBGP peer while it is left unchanged when advertising a route to an iBGP peer, and the addition of the local autonomous system in the AS Path are illustrated using Figure 1.6.

In Figure 1.6, the 10.1.1.0/24 prefix originates on router A with an empty AS Path list and a BGP next hop of router A. Router A then advertises this prefix to router B. Router B, when advertising the route to router C, adds AS65100 to the AS Path list and sets the BGP next hop to 10.1.3.1, because router C is an exterior peer (a peer outside the autonomous system). Router C then advertises the 10.1.1.0/24 prefix to router D without changing the AS Path or the BGP next hop, since

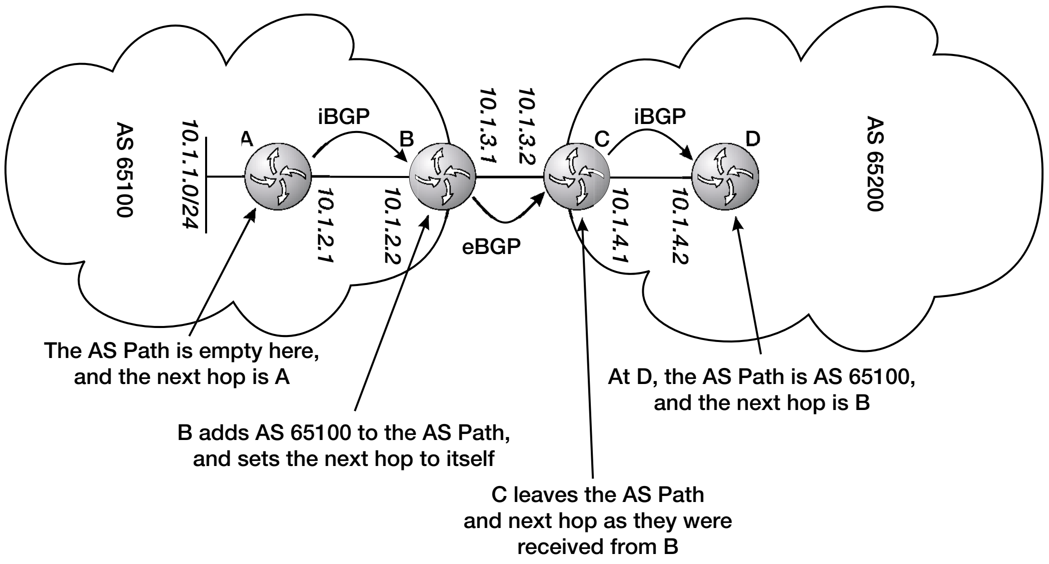
AS 65100
10.1.1.0/24
A
iBGP
B
10.1.2.1
10.1.2.2
10.1.3.1
10.1.3.2
eBGP
C
iBGP
D
10.1.4.1
10.1.4.2
AS 65200
The AS Path is empty here, and the next hop is A
B adds AS 65100 to the AS Path, and sets the next hop to itself
C leaves the AS Path and next hop as they were received from B
At D, the AS Path is AS 65100, and the next hop is B

Figure 1.6
eBGP and iBGP peering.

router D is an interior peer (a peer within the same autonomous system). Router D will need a path to router B in order to consider this prefix reachable; generally, the BGP next hop reachability information is provided by advertising the link between B and C through an interior gateway protocol, or through iBGP, originating the link as a prefix from C into AS65100.

All BGP peers are connected over a TCP transport session. As such, IP reachability must exist before a pair of BGP speakers can peer with one another. For iBGP sessions, reachability between speakers typically is provided using an interior gateway protocol. EBGP peers are normally directly connected over a single hop (across a single link), with no intervening routers, and therefore require no additional underlying routing information. There are mechanisms for connecting eBGP peers across multiple hops; these are covered in more detail in Multipath section of Chapter 7.

Converting an understanding of BGP into practical, running configurations isn't always as easy at it seems, so we will often provide sample configurations for networks used as examples. These examples will be shown using Cisco IOS Software as the operating system. For the network in Figure 1.5, the following configurations, along with some explanation of the various parts of the configuration, are provided.

```
!
hostname router-a
!
router BGP 65100
! enables the BGP process and defines the local AS number
 network 10.1.1.0 mask 255.255.255.0
 ! the above line causes router-a to originate the 10.1.1.0/24
 ! prefix in BGP
 neighbor 10.1.2.2 remote-as 65100
 ! configures an iBGP session with router-b

!
hostname router-b
!
router bgp 65100
 ! The number following the router bgp command above is
 ! the local autonomous system number
 neighbor 10.1.2.1 remote-as 65100
 ! configures an iBGP session with router-a
 neighbor 10.1.3.2 remote-as 65200
 ! configures an eBGP session with router-c; note the AS
 ! number in this command does not match the AS number of
 ! the local router

!
hostname router-c
!
router bgp 65200
 neighbor 10.1.3.1 remote-as 65100
 ! configures an eBGP session with router-b; note the AS
 ! number in this command does not match the AS number of
 ! the local router
 neighbor 10.1.4.2 remote-as 65200
 ! configures an iBGP session with router-d
 network 10.1.3.0 mask 255.255.255.0
 ! configures this router to advertise the 10.1.3.0/24
 ! prefix to router-d, so router-d will be able to reach the
 ! BGP nexthop towards 10.1.1.0/24; reachability could also
 ! be provided through an interior gateway protocol or static
 ! routing

!
hostname router-d
!
router bgp 65200
 neighbor 10.1.4.1 remote-as 65200
 ! configures an iBGP session with router-c
```

With these configurations in place, router D should learn the 10.1.1.0/24 prefix from router C, and install it as a reachable destination within its routing table.

BGP Notifications

Throughout the duration of a BGP session between two BGP speakers, it's possible that one of the two peers will send some data in error or send malformed data or data the other speaker doesn't understand. The easiest remedy in any of these situations is simply to shut down the BGP session, but a simple session shutdown doesn't provide any diagnostic information to the speaker that transmitted the information that triggered the peering session to shut down, and therefore no corrective action can be taken. To provide the information needed to take corrective action, BGP includes *Notifications,* which should be sent by the BGP speaker closing the session.

Notifications consist of three parts:

- A notification code
- A notification subcode
- A variable-length data field

The Notification code indicates what type of error occurred:

- An error occurred in a message header, error code 1.
- An error occurred in the Open message, error code 2.
- An error occurred in an Update message, error code 3.
- The hold timer expired, error code 4.
- An error occurred in the finite state machine, error code 5.
- Cease, error code 6.

The subcode provides more information about the error–for instance, where in the Open message the error was. The BGP speaker transmitting the Notification can fill in the data field with information such as the actual part of the Open message causing the error. While the data field is variable in length, there is no length field in the Notification code format. This is because the length of the data field is implied by the length of the complete message.

Message Header Errors

Message header errors generally indicate problems in the packet format. Since TCP is a reliable transport service, message header errors should be very rare, although it is possible for an implementation of BGP to malform a packet, causing this type of error. Three subcodes are defined in the base BGP specification:

- Connection not synchronized
- Bad message length
- Bad message type

Open Message Errors

Notifications transmitted while two BGP peers are opening a session are generally the result of misconfiguration rather than packet-level errors or problems in a BGP implementation.

- Unsupported version number, which means the BGP peer has transmitted a BGP version this speaker does not support.
- Bad peer autonomous system; the peer has claimed an autonomous system number that isn't valid.
- Bad BGP Identifier; the peer has transmitted a BGP router ID that is invalid.
- Unsupported optional parameter; the peer has indicated it wants to use some optional parameter the receiver doesn't support.
- Authentication failure; the peer is sending packets that are encrypted or authenticated in some way, but the authentication check is failing.
- Unacceptable hold time.

Update Message Errors

As BGP peers exchange updates, a number of errors can occur that make it impossible for one speaker to process an update transmitted by the other speaker:

- Malformed attribute list; the list of attributes included in the update packet has some error that makes it unreadable by the receiver.
- Unrecognized well-known attribute; the sender is including an attribute the receiver must be able to process but does not recognize.
- Missing well-known attribute; the sender is not including a required well-known attribute.
- Attribute flags error; the flags included with an attribute are not formed correctly (generally flags carry various options that apply to the attribute).
- Attribute length error; an attribute is either too long or too short.

- Invalid Origin; the origin code attribute is set to an invalid value.
- Invalid Next Hop; the Next Hop attribute is set to an invalid value.
- Optional attribute error; an optional attribute is malformed.
- Invalid network field; a prefix included in the update is invalid.
- Malformed AS Path; the AS Path included in the update is invalid.

Cease

The Cease code indicates to the receiver that the peer for some reason has chosen to close the BGP connection. The Cease Notification is not sent if a fatal error occurs, but rather it provides a graceful mechanism to shut down a BGP connection.

BGP Capabilities

There are various extensions to BGP that, to function correctly, require support of both BGP speakers in a session. How does a BGP speaker know when another BGP speaker it's peering with supports these extensions to BGP? Through BGP capabilities, which are negotiated when a BGP session is started.

The ability for one BGP speaker to advertise capabilities to a peer BGP speaker is described in RFC3392, Capabilities Advertisement with BGP-4. *draft-ietf-idr-dynamic-cap* describes a way in which these capabilities can be advertised dynamically not only on session startup but after a session is established.

When first initiating a session, a BGP speaker sends an Open message describing various parameters, including a set of *capability codes,* one for each optional capability it supports. Capability codes are defined for things such as

- Route refresh, capability code 0 and 2
- Multiprotocol extensions, capability code 1

- Cooperative route filtering, capability code 3
- Dynamic capability exchange, capability code 6
- Graceful restart, capability code 64
- Four octet autonomous system numbers

The applicability and value of these and other BGP capabilities and extensions will be discussed in later sections.

If a BGP speaker receives a capability code it does not support when enabling a peering with another BGP speaker, it will send a Notification message to its peer, which shuts down the session, with a notification subcode indicating that the peer requested a capability the local BGP speaker doesn't support. The receiving peer can either break off communications on receipt of a notification code indicating an unsupported capability, or it can attempt to peer again without that capability enabled.

The BGP Peering Process

There are a lot of elements to the BGP peering process; when a BGP speaker begins a session with a new peer, it must determine if it is peering with an external neighbor or an internal neighbor, it must negotiate capabilities, and it must do a number of other things. The BGP session state machine in Figure 1.7 illustrates the process in an attempt to bring all these different actions together in one place.

BGP Attributes

BGP attributes are a confusing array of information carried in a BGP update capable of indicating anything from path preference to various additional pieces of information about a route, either within an autonomous system or outside an autonomous system. There are four basic types of attributes:

- **Well-known mandatory attributes**; these attributes must be recognized by all BGP speakers and must be included in all update messages. Almost all of the attributes impacting the path decision process, described in the next section, are well-known mandatory attributes.

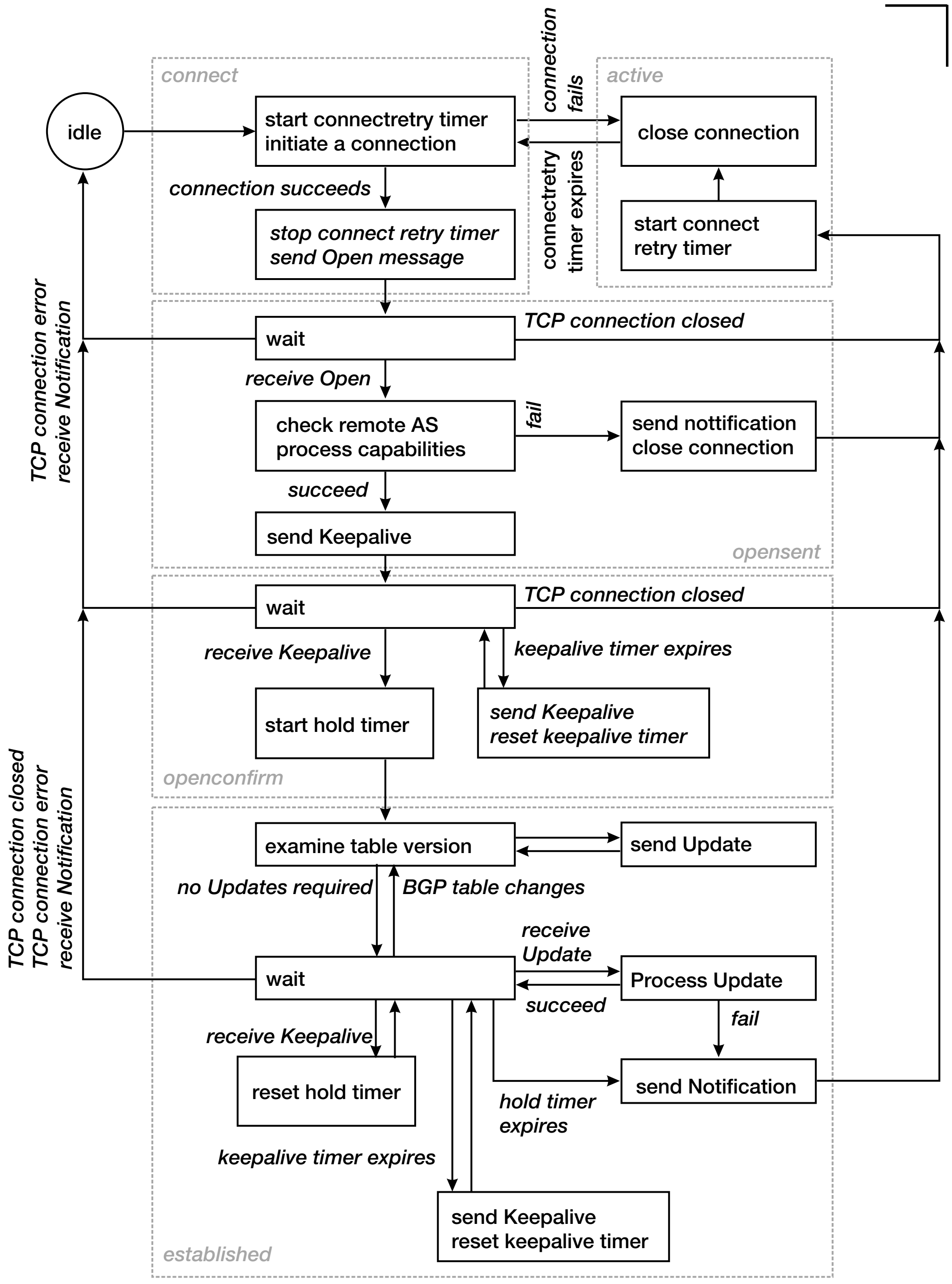

Figure 1.7
The BGP peering state machine.

- **Well-known discretionary attributes**; these attributes must be recognized by all BGP speakers and may be carried in updates but are not required in every update.
- **Optional transitive attributes**; these attributes may be recognized by some BGP speakers, but not all. They should be preserved and advertised to all peers whether or not they are recognized.
- **Optional nontransitive attributes**; these attributes may be recognized by some BGP speakers, but not all. If an update containing an optional transitive attribute is received, the update should be advertised to peers without the unrecognized attributes.

Figure 1.8 illustrates the way in which attributes are included in a BGP update message.

There are several other attributes not shown in Figure 1.8 but included in BGP, such as Communities and Extended Communities.

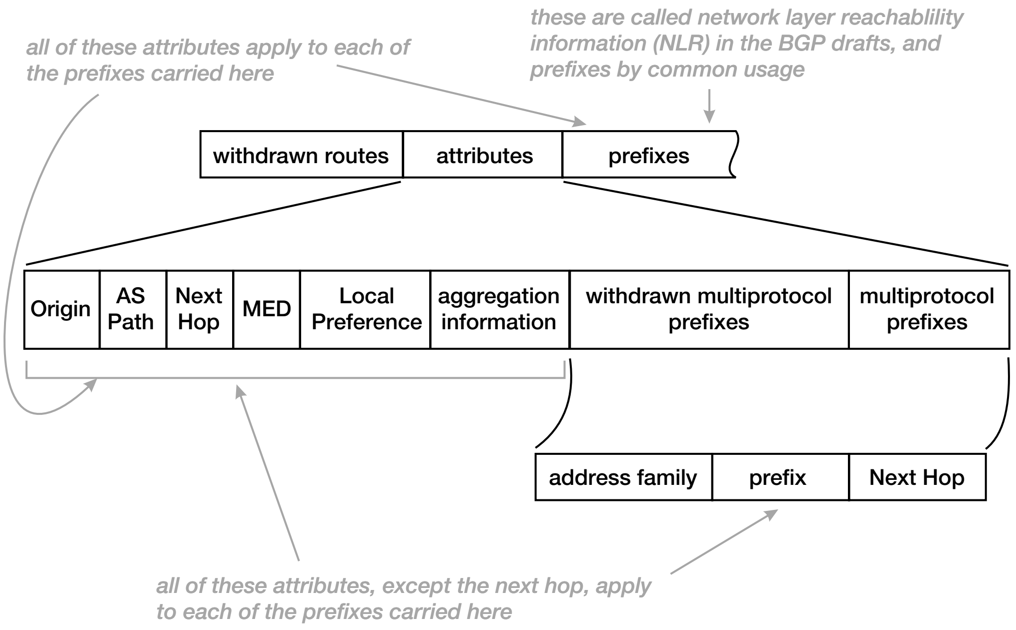
these are called network layer reachablility information (NLR) in the BGP drafts, and prefixes by common usage
all of these attributes apply to each of the prefixes carried here
withdrawn routes
attributes
prefixes
Origin
AS Path
Next Hop
MED
Local Preference
aggregation information
withdrawn multiprotocol prefixes
multiprotocol prefixes
address family
prefix
Next Hop
all of these attributes, except the next hop, apply to each of the prefixes carried here

Figure 1.8
Carrying attributes within a BGP update.

Origin Code

The ORIGIN is a well-known mandatory attribute that indicates the origin of the prefix or, rather, the way in which the prefix was injected into BGP. There are three origin codes, listed in order of preference:

- IGP, meaning the prefix was originated from information learned from an interior gateway protocol
- EGP, meaning the prefix originated from the EGP protocol, which BGP replaced
- INCOMPLETE, meaning the prefix originated from some unknown source

The following configurations illustrate two of these origin codes using Cisco IOS Software.

```
!
hostname router-a
!
....
!
interface Ethernet1/0
 ip address 10.1.12.4 255.255.255.0
!
....
!
interface Serial3/0
 ip address 10.0.7.4 255.255.255.0
!
....
!
router bgp 65500
 no synchronization
 bgp log-neighbor-changes
 network 10.0.10.0
 redistribute static metric 10
 neighbor 10.0.7.10 remote-as 65501
 no auto-summary
!
ip classless
ip route 10.7.7.0 255.255.255.0 10.1.12.1
```

```
!
hostname router-b
!
....
!
interface Serial0/0
 ip address 10.0.7.10 255.255.255.0
!
....
!
router bgp 65501
 no synchronization
 bgp log-neighbor-changes
 neighbor 10.0.7.4 remote-as 65500
 no auto-summary
!

router-b#sho ip bgp
BGP table version is 3, local router ID is 10.0.16.10
Status codes: s suppressed, d damped, h history, * valid, > best,
i -internal, r RIB-failure
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop       Metric  LocPrf Weight Path
*> 10.7.7.0/24      10.0.7.4           10       0  65500 ?
*> 10.0.10.0        10.0.7.4            0       0  65500 I
```

An Origin code of IGP typically suggests that the route was cleanly derived inside the originating AS. An Origin code of EGP suggests that the route was learned via the EGP protocol. Origin codes of Incomplete typically result from aggregation, redistribution, or other indirect ways of installing routes into BGP within the originating AS.

AS Path

The AS_PATH is a well-known mandatory attribute, and as described in the section BGP Path Vector Implementation earlier in this chapter, is the list of all autonomous systems the prefixes contained in this update have passed through. The local autonomous system number is added by a BGP speaker when advertising a prefix to an eBGP peer.

Next Hop

The BGP NEXT_HOP is a well-known mandatory attribute. As described in the section Interior and Exterior Peering earlier in this chapter, the Next Hop attribute is set when a BGP speaker advertises a prefix to a BGP speaker outside its local autonomous system (it may also be set when advertising routes within an AS; this will be discussed in later sections). The Next Hop attribute may also serve as a way to direct traffic to another speaker, rather than the speaker advertising the route itself, as Figure 1.9 illustrates.

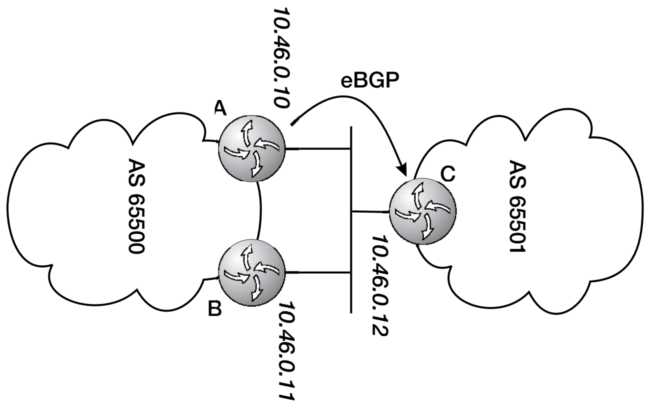

Figure 1.9
BGP third party Next Hop.

The following configurations from a router running Cisco IOS Software illustrate Router C using B as the BGP next hop for destinations in AS65500, even though Router C is learning these routes directly from A.

```
!
hostname router-a
!
....
!
interface FastEthernet0/1
 ip address 10.46.0.10 255.255.255.0
 duplex auto
 speed auto
!
....

!
```

```
router bgp 65500
 no synchronization
 bgp log-neighbor-changes
 network 10.46.12.0
 neighbor 10.46.0.12 remote-as 65501
 neighbor 10.46.0.12 route-map setnexthop out
 no auto-summary
!
....
!
access-list 10 permit 10.46.12.0
!
route-map setnexthop permit 10
 match ip address 10
 set ip next-hop 10.46.0.11
!

!
hostname router-b
!
....
!
interface FastEthernet0/1
 ip address 10.46.0.12 255.255.255.0
 duplex auto
 speed auto
!
....

!
router bgp 65501
 no synchronization
 bgp log-neighbor-changes
 neighbor 10.46.0.10 remote-as 65500
 no auto-summary
!

router-b#show ip bgp
BGP table version is 2, local router ID is 208.0.14.12
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal, r RIB-failure
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.46.12.0       208.0.0.11               0             0 65500
```

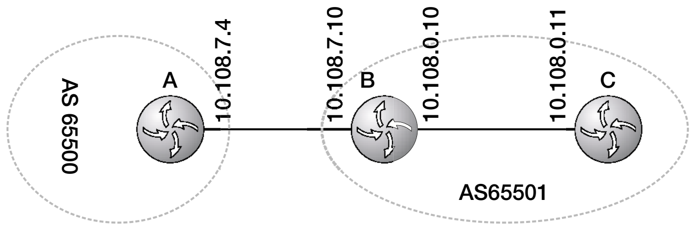

Figure 1.10
Setting the Next Hop to Self in iBGP.

Most BGP implementations deployed today also allow the network administrator to set the BGP next hop when advertising a route between iBGP peers; Figure 1.10, and the following configuration from a router running Cisco IOS Software, illustrates this fact.

```
!
hostname router-a
!
....
!
interface Serial3/0
 ip address 10.108.7.4 255.255.255.0
!
....
!
router bgp 65500
 no synchronization
 bgp log-neighbor-changes
 network 10.108.12.0 mask 255.255.255.0
 neighbor 10.108.7.10 remote-as 65501
 no auto-summary
!

!
hostname router-b
!
....
!
interface Serial0/0
 ip address 10.108.7.10 255.255.255.0
```

```
!
....
!
interface FastEthernet0/1
 ip address 10.108.0.10 255.255.255.0
 duplex auto
 speed auto
!
....
!
router bgp 65501
 no synchronization
 bgp log-neighbor-changes
 neighbor 10.108.0.11 remote-as 65501
 neighbor 10.108.0.11 next-hop-self
 neighbor 10.108.7.4 remote-as 65500
 no auto-summary
!

router-b#show ip bgp
BGP table version is 2, local router ID is 10.108.16.10
Status codes: s suppressed, d damped, h history, * valid, > best,
              i - internal, r RIB-failure
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.108.12.0/24   10.108.7.4               0             0 65500 i

!
hostname router-c
!
....
!
interface FastEthernet0/1
 ip address 10.108.0.11 255.255.255.0
 duplex auto
 speed auto
!
....
!
router bgp 65501
 no synchronization
 bgp log-neighbor-changes
 neighbor 10.108.0.10 remote-as 65501
 no auto-summary
!

router-c#show ip bgp
BGP table version is 2, local router ID is 10.108.13.11
Status codes: s suppressed, d damped, h history, * valid, > best,
              i - internal, r RIB-failure
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i10.108.12.0/24   10.108.0.10              0    100      0 65500 i
```

The reason why a network administrator would want to do this is discussed in later sections.

Multiple Exit Discriminator (MED)

The MUTLI_EXIT_DISC (MED) is an optional nontransitive attribute that provides a mechanism for the network administrator to convey to adjacent autonomous systems to optimal entry point in the local AS; Figure 1.11 illustrates this concept.

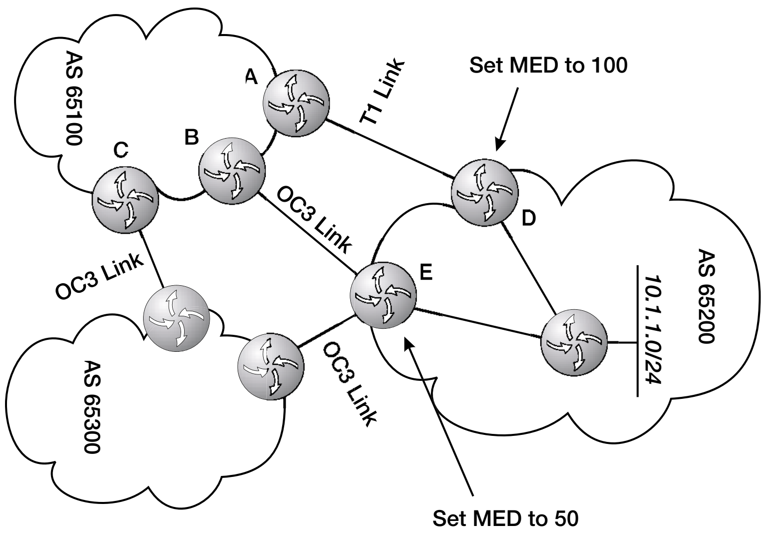

Figure 1.11
The Multiple Exit Discriminator.

Here, AS 65200 is setting the MED on its T1 exit point to 100 and the MED on its OC3 exit point to 50, with the intended result that the OC3 connection be preferred. However, the problem with using the MED in this way becomes apparent with this simple example. First, AS 65100 will receive three paths to 10.1.1.0/24, one through AS 65300 and two through AS 65200. The MED of the path through AS 65100 and the paths through AS 65200 will not be compared since their AS Path is not the same. If AS 65100 has set its BGP local preferences on router A, B, and C, to favor the path through AS 65300, then the MED from AS 65200 will have no impact as MED is considered after local preference in the BGP decision algorithm.

MEDs received from different autonomous systems are not compared as a default behavior, though many implementations provide a mechanism to enable comparing of MEDs between different autonomous systems. Benefits and offshoots of using MEDs and comparing them between different AS Paths will be discussed in later sections.

If the path through AS 65300 did not exist, or was not preferred over the path through AS 65200 for some other reason, the MEDs advertised by routers D and E might have some impact on the best path decision made by AS 65100. However, if AS 65100 sets some BGP metric with a higher degree of preference in the decision algorithm, such as the local preference, to prefer one path over the other, the MED would never be considered.

Local Preference

The LOCAL_PREF attribute is a well-known attribute that represents the network operator's degree of preference for a route within the entire AS. The larger the value of the local preference, the more preferable the route is; Figure 1.12 illustrates.

AS 65100 is receiving two possible paths to the 10.1.1.0/24 network, one of which is received through AS 65200 and the other of which is received through AS 65300. Although the path through AS 65200 is shorter—one AS hop rather than two—AS 65100's network administrator would prefer to send traffic destined to this prefix along the high-speed outbound OC3 connection rather than along the outbound T1. Setting the local preference on this prefix as it is received on router

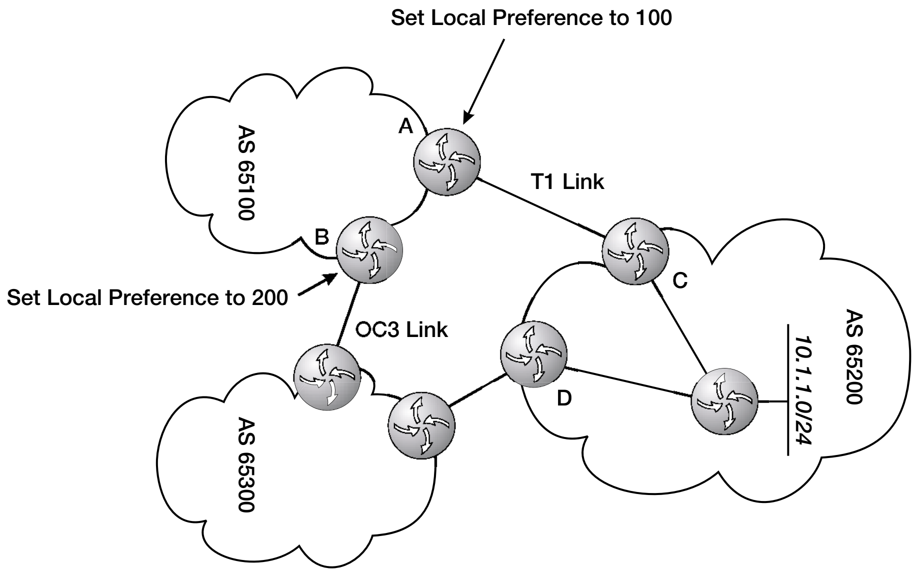

Figure 1.12
Local Preference.

A to 100, and on router B to 200, causes all of the BGP speakers within AS 65100 to prefer the path through B, thus preferring the higher-speed link.

Communities

The COMMUNITIES attribute is an optional transitive attribute. Communities are, effectively, tags that can carry almost any information about a route within or between autonomous systems. Communities are used to group routes sharing common characteristics that cannot be described using the other attributes. Communities generally are not directly used to determine policy or the best path to a destination. That is, while a community itself does not influence the BGP route selection algorithm, communities are typically used to trigger underlying policies that take effect based on the value of the associated community (e.g., communities can be used to match and modify one or more of the BGP attributes that do impact the results of the best path selection algorithm).

Several communities are defined as well-known, or global, communities, which should be recognized by all BGP implementations:

- The NO_EXPORT community, which states that the group of routes marked with this community should not be advertised outside of the local autonomous system.
- The NO_ADVERTISE community, which states the group of routes marked with this community should not be advertised to any BGP peer of the speaker that receives it.
- The NO_EXPORT_SUBCONFED community, which states the group of routes marked with this community should not be advertised outside a single autonomous system, even if that autonomous system is a part of a confederation of autonomous systems.

Communities are 32 bits (4 octets) long, with the following standards for using the community space:

- The communities numbered 0x00000000 through 0x0000FFFF and 0xFFFF0000 through 0xFFFFFFFF are reserved for future assignment of well-known communities.
- The recommended encoding for all other communities is the two-octet autonomous system number of the AS that attached the community to the route in the first two octets of the community number. The remaining two octets can be assigned based on policies internal to the AS.

RFC1997, BGP Communities Attribute, describes communities within BGP. RFC1998, An Application of the BGP Community Attribute in Multi-Home Routing, describes the use of the NO_EXPORT community in dual-homed environments, and the section Conditional Communities in Chapter 7 describes an extension of RFC1998.

Extended Communities

BGP extended communities, as their name implies, are an extension to BGP communities. The primary differences between communities and extended communities are as follows:

- Extended communities are 64 bits, or 8 octets, in length.
- The extended community number space is more structured than the standard community address space, as described next.

Figure 1.13 illustrates the extended community layout.

If the I bit is clear (0), the community type was assigned on a “first come, first served” basis through the Internet Assigned Numbers Authority (IANA). If the I bit is set (1), the community type is either experimental or was assigned through IETF consensus. If the type code indicates that a subtype (or low type) is included, the data field is 6 octets in length. If the type code indicates that a subtype is not included, the data field is 7 octets in length.

Extended communities defined in various drafts include the following:

- Autonomous System Specific (two octet), type 0x0 (or 0x4), which allows the local network administrator to carry communities specific to their autonomous system by setting the subtype to a value indicating the type of information being carried, the first two octets of the data portion to their autonomous system number, and the remaining four octets to the data carried.
- Autonomous System Specific (four octet), type 0x02 (or 0x42); this extended community is similar to the AS Specific extended community, except it allows four octets from the data field for the autonomous system number and two octets for the data carried.
- IPv4 Address Specific Type, type 0x01 or 0x41, which allows the owner of an IPv4 address block to encode some information in an extended community pertinent to this address space. The

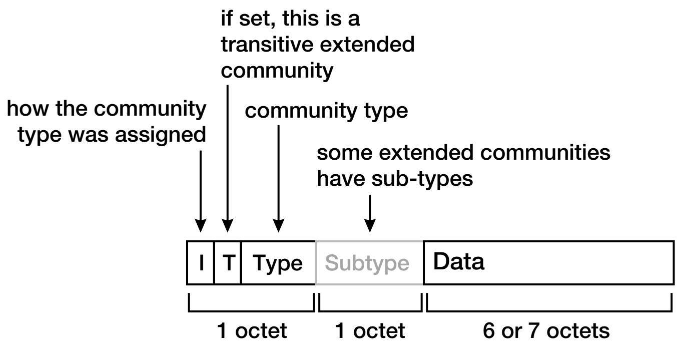
if set, this is a transitive extended community
how the community type was assigned
community type
some extended communities have sub-types
I
T
Type
Subtype
Data
1 octet
1 octet
6 or 7 octets

Figure 1.13
The extended community layout.

subtype field is set by the originator to indicate the type of information carried, the first four octets of the data field are set to the IPv4 address, and the last two octets of the data field are set to the pertinent information.

- Opaque, type 0x03 or 0x43, allows opaque data to be carried within an extended community. The subtype field is set to a value set according to consensus within the IETF or other Internet Addressing and Number Authority (IANA) rules.
- Route Target, which is a subtype of either the two octet AS specific, four octet AS specific, or IPv4 specific types, with the subtype set to 0x02. This extended community is described in further detail in Chapter 10, Deploying BGP and MPLS VPNs.
- Route Origin, which is a subtype of either the two octet AS specific, four octet AS specific, or IPv4 specific types, with the subtype set to 0x03. This extended community is described in further detail in Chapter 10, Deploying BGP and MPLS VPNs.
- Link Bandwidth, which is a subtype of the two octet AS specific type, with a subtype of 0x4. This community is described in more detail in Chapter 7, New Features in BGP.

BGP extended communities are described in the Internet Draft document, draft-ietf-idr-bgp-ext-communities, *BGP Extended Communities Attribute*, which should be progressed to RFC status sometime in the near future.

Multiprotocol Addresses

The original BGP packet format was formatted around IPv4 addresses, which are 4 octets in length. In order to carry new types of addresses, such as IPv6, MPLS Labels, VPN information, CLNS addresses, and others, special address family attributes were created to carry these address types. Each type of address is identified using an *Address Family Identifier* (AFI), and a *Subsequent Address Family Identifier* (SAFI). The ability of BGP to carry multiple address types is used in carrying MPLS VPN information, as described in Chapter 10, Deploying BGP and MPLS VPNs. This capability is also used to carry CLNS and IPv6 address, as described in the section Multiprotocol BGP, below.

The ability to carry multiple address types and other information pertaining to virtual private networks, is outlined in the IETF Internet Draft draft-ietf-idr-rfc2858bis, which is currently in draft state and specifies an update and intended to oboselete RFC 2858.

Attributes and Aggregation

Aggregation, or summarization, not only hides reachability information, it also hides topology information. In BGP, this means hiding the AS Path and other attributes of the prefixes aggregated.

Aggregation and the AS Path

Figure 1.14 illustrates the interaction between the AS Path and aggregation.

In this network, routers C and D are advertising 10.1.2.0/24 and 10.1.3.0/24, respectively, to router B, which is in another autonomous system. Router B is aggregating these two advertisements toward router A, advertising the single prefix 10.1.2.0/23. But how does router B build the AS Path in the route it advertises to router A?

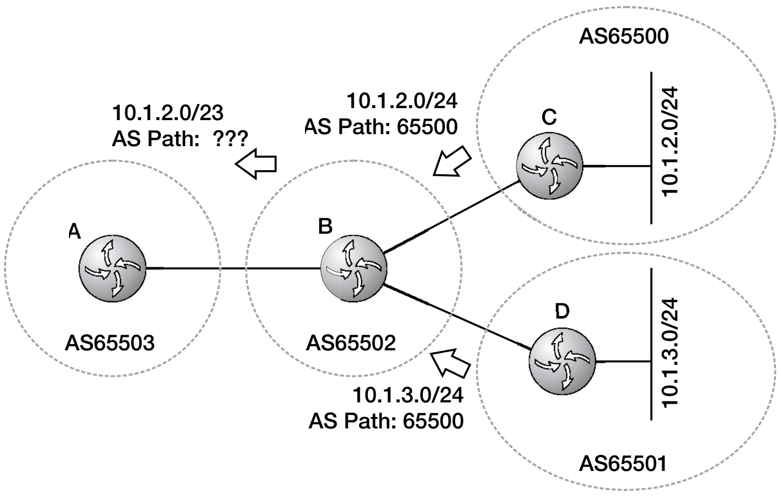

Figure 1.14
Aggregation's impact on route attributes.

It can't act as though AS65500 and AS65501 don't exist, since that would break the inherent loop detection qualities of the AS Path, and it would also break any policies based on the AS Path containing AS65500 or AS65501 in AS65503 (or downstream). It can't include both of these autonomous systems in the AS Path sequentially, since that would imply that the path to reach either of these networks passes through both of these autonomous systems.

The solution to this problem is to include both of the originating autonomous systems as an *AS Set*. An AS Set includes a set of autonomous systems possibly included in the path to a given advertised route, in no particular order. When advertising this aggregate, then router B would advertise (65502 {65500, 65501}) in the AS Path, grouping AS65501 and AS65500 into an AS Set, and prepending the local AS number, AS65500.

The Atomic Aggregate

Suppose we change the network slightly, so it now looks like the network in Figure 1.15.

Suppose router B is now receiving both 10.1.2.0/23 and 10.1.3.0/24; note that these two prefixes overlap. Router B only wants to advertise 10.1.2.0/23 toward A. Since it already has this prefix in its

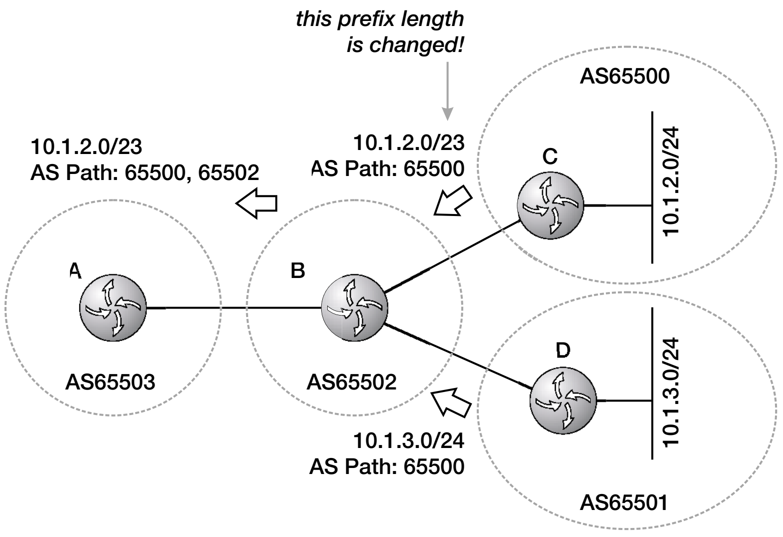

Figure 1.15
The Atomic Aggregate bit.

table, it can simply advertise the prefix as it was received from AS65500. But this leaves out the information that part of this prefix, 10.1.3.0/24, is actually reachable in AS65501; in fact, AS65501 doesn't appear in the AS Path of the advertisement to router A.

In this case, router B should include the *Atomic Aggregate* attribute in the advertisement. This tells router A that while the AS Path, as presented, is loop free, there is some longer prefix length component within this prefix reachable through an autonomous system not listed in the AS Path. The Atomic Aggregate is a well-known discretionary attribute.

BGP's Best Path Algorithm

Interior Gateway Protocol Cost

BGP can also take into consideration the cost of reaching the exit point from within the autonomous system (Figure 1.16).

At router C, the two paths advertised from AS 65200, and passed via iBGP through routers A and D, look the same. From C's perspective, the cost of transmitting the packet from the exit point to the final destination is the same, so it only makes sense to take the shortest path to the exit point from the AS. The easiest way to determine the shortest path to the exit point of the AS is to use interior gateway protocol metrics to compare the paths.

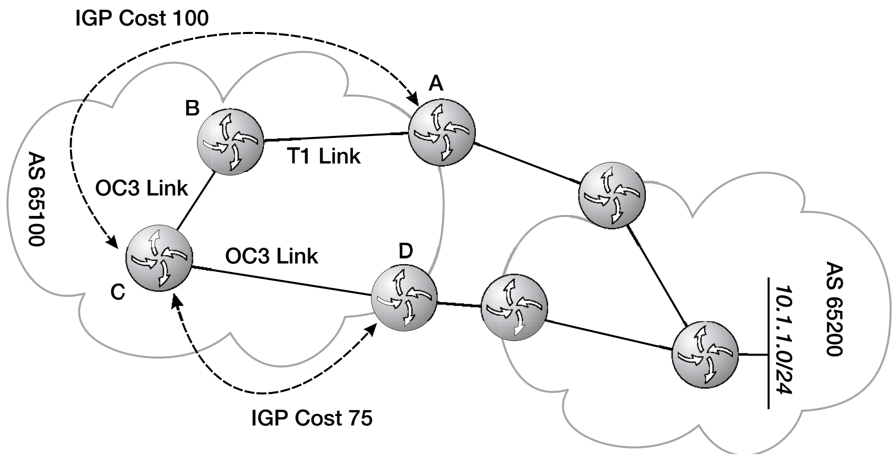

Figure 1.16
Interior Gateway Protocol cost in BGP metrics.

The IGP metric check assumes that the same IGP is running throughout the AS; if several different IGPs are running within the same AS, their metrics are not comparable, and the results of this check can actually produce suboptimal routing to the edge of the AS.

IGP metric allocation values are typically derived from a number of factors. For example, the number of "route miles" or "propagation delay" a particular connection uses may be factored. Other factors could include the available capacity of a link (e.g., a 1000-Mbps gigabit Ethernet connection would be 10 times more preferable than a 100-Mbps Fast Ethernet connection), or perhaps some multiple is used to reflect the reliability of the Physical Layer medium. The method used to derive IGP metrics varies widely among network operators and is influenced by many other factors as well (e.g., the available metric allocation range provided by the interior gateway protocol being used).

BGP Identifier

The BGP Identifier is a four-octet value exchanged with BGP peers in the Open message. The value of the BGP Identifier is determined on startup and is the same for all peers of the BGP speaker. If the value of the BGP Identifier changes all the BGP sessions must be reset and must be reinitiated using the new value. As such, the local BGP Identifier value is typically derived from an IP address associated with a loopback or other similar virtual system interface, so as to avoid potential instability introduced by interface or other hardware failures. Note that it is not a requirement that the value be derived from an active IP address on the local system, or that if it was, the address must remain active. However, the value must be unique within the routing system and not deriving the BGP Identifier value from an active system IP address may introduce conflicts. Typically, implementations first attempt to get the value from a virtual interface, though the value is often determined by using a value of the lowest IP address configured on any active interface on the BGP speaker.

The BGP Identifier is often referred to simply as router ID. The router ID of the advertising router is generally considered the "tie breaker" in the BGP bestpath algorithm; if multiple paths are identical in cost, using the router ID to break the tie allows the routing decision to be made deterministically throughout the autonomous system (hence the requirement for uniqueness of the value). If two paths have the same preferences throughout all of the other steps, the path through the advertising router with the lowest router ID will be considered the best path, installed in the local routing table, and advertised to peers.

Weight

Most implementations of BGP provide a mechanism that allows the network administrator to set the weight on a particular prefix such that it wins the local best path calculation without impacting the best path calculations used in the remainder of the network; this factor is normally referred to as the weight, and influences decisions only by the local router. Weight is not a defined BGP metric, and is not advertised as an attribute to a route in BGP, so it does not impact the routing decision at any other router in the network.

Note that caution should be taken when manipulating weight as it only impacts path selection on the local system and may result in routing loops.

Review Questions

1. What is a routing domain from BGP's perspective? How is this different from a routing domain within IS-IS?
2. What are the two primary differences between an interior gateway protocol and an external gateway protocol?
3. What types of policies would you normally see implemented through BGP?
4. For what does BGP use the path information it carries through the network?
5. Why does BGP treat each autonomous system as a point on the connectivity graph? What does this imply about BGP's usefulness within an autonomous system?
6. What transport does BGP use to build a session to another BGP speaker? What local port number and remote port number does BGP use when initiating a connection?

7. How is a collision resolved between two BGP speakers attempting to open a connection at the same time?
8. Define *prefix, NLRI,* and *attribute.*
9. How many sets of attributes can a single BGP update contain? How many prefixes?
10. What are the four primary differences between eBGP peering relationships and iBGP peering relationships?