DEPLOYING JUNIPER DATA CENTERS WITH EVPN VXLAN

ANINDA CHATTERJEE

FREE SAMPLE CHAPTER

The depth, detail, and thoroughness of this book easily surpasses any other VXLAN/EVPN book on the market. And it is the only book available that covers the topic from a Juniper Junos and an Apstra perspective. Whether you want a VXLAN/EVPN technical deep-dive, want to learn how to configure it on Junos, want to learn Apstra's Intent-Based Networking platform, or are studying for your JNCIE-DC lab, this book is essential for data center engineers and architects.

—Jeff Doyle, Director of Solutions Architecture
Juniper Networks/Apstra

Aninda has written the new definitive guide for learning, building and operating EVPN networks. This book should be on the shelves of any network engineer, from NOC technicians to senior architects.

—Pete Lumbis, CCIE No. 28677, CCDE 2012:3

Today's data centers require modern technologies that simplify operations and assure reliability at the tremendous scale demanded by AI training and digital applications. Juniper innovation is in the forefront with Apstra Intent-Based Networking automation for EVPN VXLAN multivendor networks. *Deploying Juniper Data Centers with EVPN VXLAN* is a comprehensive guide that includes all these technologies in one place to understand how they work together for robust, automated DC operations. Architects and operators responsible for the integrity of the data center will want this go-to book to advise step by step how to set up and run their network following Juniper recommended, best practice designs, tools, and workflow.

—Mansour Karam, GVP
Juniper Networks

Juniper's data center fabric solutions are world-renowned for their completeness and quality. This book begins right at the beginning, with basic data center fabric design, BGP in the data center, and VXLAN. After covering these topics, Aninda moves into an explanation of Apstra, one of the most complete multi-vendor intent-based data center fabric systems.

The many graphics and screen shots, combined with the detailed configuration and sample outputs, provide designers and operators alike with deeply researched and well-explained information about building and operating a data center fabric using Juniper hardware and software.

I even learned a few things about Apstra reading through this book—although I have built and operated networks using Apstra's technology.

I highly recommend this book for engineers looking for a good explanation of Juniper data center solutions.

—Russ White

Aninda is an outstanding engineer with an insatiable thirst for knowledge and discovery. His drive is endless and a wonderful opportunity for himself and many others to learn and explore subjects and technologies, as he is able to simplify them in a way that allows others to learn seamlessly. I have enjoyed Aninda's content for several years now. He has contributed [to] the community through webinars, articles, white papers, and blogs, which makes his book a logical step to consolidate his contributions and knowledge.

Aninda's work will always have my support and endorsement.

—David Penaloza, Principal Engineer

Deploying Juniper Data Centers with EVPN VXLAN

Deploying Juniper Data Centers with EVPN VXLAN

Aninda Chatterjee

Addison-Wesley

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2024934201

Published by:

Addison-Wesley

Hoboken, New Jersey

ISBN-13: 978-0-13-822539-1

ISBN-10: 0-13-822539-7

$PrintCode

GM K12, Early Career and Professional Learning: Soo Kang

Executive Editor: Brett Bartow

Development Editor: Ellie C. Bru

Managing Editor: Sandra Schroeder

Senior Project Editor: Tonya Simpson

Copy Editor: Bill McManus

Indexer: Erika Millen

Proofreader: Donna E. Mulder

Compositor: codeMantra

Technical Reviewers: Jeff Doyle, Vivek Venugopal, and Ridha Hamidi

Editorial Assistant: Cindy Teeters

Cover Designer: Chuti Prasertsith

Figure Credits

Figures 3.11–3.15, 5.7, 5.8, 5.12, 5.13, 5.21, 5.22, 5.28, 5.30, 5.32, 5.33, 5.35, 5.47, 5.52–5.54, 6.7, 6.12, 6.13, 6.17, 7.8, 7.13, 7.14, 7.19, 7.22, 7.24, 7.26, 8.11, 8.13, 8.16, 8.21, 8.22, 8.26, 8.28, 9.5, 9.10, 10.4, 10.6, 10.8, 10.9, 10.12–10.14, 10.18, 10.19, 11.10, 11.17, 11.18, 11.21, 11.27–11.30, 11.34, 11.38–11.40, 12.15–12.18, 13.9, 13.40, 13.41: Wireshark Foundation

Figures 12.1–12.4, 12.6, 12.8, 12.10, 12.13, 12.14, 12.19–12.28, 12.30–12.40, 12.42–12.65, 12.67–12.75, 13.2–13.8, 13.12–13.26, 13.28–13.39, 13.42, 13.43, 13.45–13.59, 14.2–14.10, 15.3–15.7, 15.11–15.16, 15.18, 15.19, 15.24–15.33: Juniper Networks, Inc

Dedications

This book is dedicated to the family I was born into, and the family I married into.

Foreword

The titans of the networking industry stand tall not because they have proven themselves masters of theory. Nor is it because they have waxed poetic about all manners of enabling our connected world. Those whose heads and shoulders rise above achieve their place because they are practitioners.

And so, as we evaluate technical works for their transformative potential, we should come to know our authors by their hands-on skills more than their willingness to pontificate. Their experience is the bedrock on which truly great works are built.

But let's be honest. Our industry is one where most of the really important work is done behind closed doors, in places where peering eyes might never reach. So how do you assess skills when the work they deliver is hidden by design?

I have had the great pleasure of building multiple organizations over the years. I have led data center businesses at multiple large vendors, which has given me the opportunity to assemble all kinds of teams. Early in my career, I would seek out experience. But as I matured and became a better leader, I learned to hunt for potential.

In my not terribly humble opinion, the highest potential exists at the intersection of capability, drive, and humility. Capability is table stakes of course, and drive is an obvious prerequisite for progress. But humility might be the secret ingredient that brings everything together.

You see, it's easy to be humble when you are starting out because you lack the experience to know how good you are. All too often, there is an inverse relationship between experience and humility—indeed, many of us become louder as we develop a stronger command over our domains! But you cannot become a true master without true humility because it is the constant awareness of what you do not know that provides the impetus to continue learning.

Naturally, our industry's strongest spokespeople will then be brimming over with humility. When Aninda and I first crossed paths, he spoke fluently about technology and experience—the kinds of things you lead with during an interview, of course. But what I heard was different. As accomplished as Aninda is, I could see that he has a real learner's mind.

That learner's mind might make for some restless nights as Aninda never seems quite comfortable with where he is in his journey. But I can't help but think of the great Theodor Seuss Geisel book *Oh, the Places You'll Go!*, because oh, what a journey it will be.

This book represents a checkpoint of sorts in Aninda's journey so far. It's meant to be an approachable guide to data center networking, explaining how EVPN VXLAN data centers are architected and operated, but importantly, using the hands-on experience that Aninda has earned through the years to make it tangible.

And if you read this book with the same learner's mind with which it has been written, oh, the places you will go.

—Michael Bushong
VP, Data Center
Nokia

Acknowledgments

As the author, it is easy to say that I wrote this book, but that is hardly the complete truth. Technically, yes, I put these words on paper, but there were so many people who helped me get to the point in my journey where I felt confident and capable enough to do this.

There are many excellent engineers who helped keep this book technically accurate, provided support when I was lost, and validated what I wrote. This also includes individuals who probably have no idea how much I have learned from them by reading their books, learning from content created by them, or have supported me in my professional and personal growth. In no particular order, they are Ridha Hamidi, Vivek Venugopal, Soumyodeep Joarder, Anupam Singh, Selvakumar Sivaraj, Wen Lin, Mehdi Abdelouahab, JP Senior, Jeff Doyle, Russ White, Jeff Tantsura, Ivan Pepelnjak, Dinesh Dutt, Pete Lumbis, Richard Michael, Peter Paluch, David Peñaloza, Daniel Dib, Naveen Bansal, Manasi Jain, and Astha Goyal.

To Brett Bartow, Eleanor Bru, Tonya Simpson, Bill McManus, Donna E. Mulder, and everyone from Pearson who helped bring this book to life: Thank you for giving me the opportunity to write this and taking a chance on a nobody. Your support through the writing, production, and composition process has been nothing short of exceptional.

To my technical reviewers, Ridha Hamidi, Vivek Venugopal, and Jeff Doyle: Thank you for reading my manuscript with gentle hands. You made it better in every way, giving constructive but honest feedback. I'd have never imagined there would come a day when I would be collaborating with Jeff Doyle, whose books I learned my networking skills from. Professional dreams do come true.

To Souvik Ghosh and Reghu Rajendran: Back in late 2011, sitting in a meeting room in the offices of Cisco Systems, Bangalore, you both interviewed me and gave me the opportunity of a lifetime. My days in Cisco TAC were some of my best. I followed you into heavy-hitting escalation roles, working together on some of the most challenging technical escalations. Thank you for guiding and mentoring me.

To Dale Miller: You are, undoubtedly, the best mentor I could have asked for. You saw potential in me when I saw none. You pushed me to new heights, to try things out of my comfort zone, and taught me what true customer advocacy means. Cisco Live conferences, bringing up new TAC centers, and solving some of the hardest escalations—we've been through it all together. You are one of the brightest spots in my career and I am glad I can call you my friend. And to Matt Esau: Like Dale, you mentored me through tough times, and even now I can reach out to you for guidance and support. I am lucky to know you and to have worked with you.

To Pete Lumbis: I can't believe we haven't worked together yet, despite literally being one "yes" away from it. You are one of the most talented engineers I have the privilege of knowing and learning from. And with all that brain power, you continue to be humble and down to earth, and you constantly reach out with helping hands. Most importantly, you genuinely look out for your peers, and you nurture those just starting this journey. You read the entire manuscript for this book, even when you had no reason to, just to give feedback and show your support.

To my dearest friends, Vivek and Gino: It's funny how long our bond has lasted because I was quite certain I was intolerable on the TAC floor, with all my cursing. But I guess like minds do think alike. We've looked out for each other since 2012. It has truly been a blessing to have both of you by my side in this journey.

To Cathy Gadecki and Mike Bushong: I have been a network engineer for over 12 years now, spanning five different roles across several companies. Your leadership, unequivocally, is the best I have experienced. For me, it wasn't about technical growth—I know how to get that for myself. You both provided personal growth and helped me nurture skills I considered irrelevant. Mike, there's no leader like you, and I don't think there ever will be. There's a reason people follow you—sure, part of it is loyalty, but there's so much more to it. You genuinely care about people and you do everything in your control to make their lives better.

To my parents, Aloke and Sujata Chatterjee, my brother, Arnab Chatterjee, and his wife, Radhika Arora: You have shaped me, as an individual, throughout my life. My interaction with the world is modeled after you and the values you taught me. Everything I have and I am stems from your kindness and love.

To my wife, Deepti: There is no measure of success without you. This last year has been grueling trying to balance work and writing this book. You were supportive every step of the way, giving me the time and space to write while managing your own work, taking care of our home, and being the best mother to our little girl. You make me a better person and a better father every day. I love you dearly and I am glad I get to walk this winding road of life with you by my side.

And to my little one, Raya: You're too young to read this, but maybe some day you will. You are the light of our lives. Now and forever.

About the Author

Aninda Chatterjee holds a Bachelor of Engineering degree in Information Science. His networking career started at AT&T, troubleshooting Layer 1 circuit issues, eventually transitioning to customer support at Cisco TAC, specializing in Layer 2. After his stint at Cisco TAC, he has held several roles across different organizations, with functions including escalation support for enterprise and data center engineering, designing, implementing, and troubleshooting enterprise and data center networks, and technical marketing for Cisco Software Defined Access (SDA).

In his current role as a senior technical marketing engineer at Juniper Networks, Aninda specializes in data center networks with EVPN VXLAN, while also focusing on the high demand of networking infrastructure for high-performance computing and AI/ML clusters.

Aninda actively writes on his personal blog, www.theasciiconstruct.com.

About the Technical Reviewers

Jeff Doyle is a director of solutions architecture at Juniper Networks. Specializing in IP routing protocols, complex BGP policy, SDN/NFV, data center fabrics, IBN, EVPN, MPLS, and IPv6, Jeff has designed or assisted in the design of large-scale IP and IPv6 service provider networks in 26 countries over 6 continents.

Jeff is the author of *CCIE Professional Development: Routing TCP/IP*, Volumes I and II; *OSPF and IS-IS: Choosing an IGP for Large-Scale Networks*; *Intent-Based Networking for Dummies*; was a co-author of *Network Programmability and Automation Fundamentals*; *Software Defined Networking: Anatomy of OpenFlow*; and is an editor and contributing author of *Juniper Networks Routers: The Complete Reference*. Jeff is currently writing *CCIE Professional Development: Switching TCP/IP*. He has also written for *Forbes*, has blogged for both *Network World* and *Network Computing*, and is co-host of the livestream show *Between 0x2 Nerds*. Jeff is one of the founders of the Rocky Mountain IPv6 Task Force, is an IPv6 Forum Fellow and a 2019 inductee into the IPv6 Internet Hall of Fame, and serves on the executive board of the Colorado chapter of the Internet Society (ISOC) and the advisory board of the Network Automaton Forum (NAF).

Vivek Venugopal has been in the computer network industry for more than 15 years. His experience spans multiple domains such as enterprise, data center, service provider networking, and network security. He has worked with a variety of networking giants such as Cisco Systems, Juniper Networks, and VMware in various capacities, and has founded a startup in the networking education space as well.

Ridha Hamidi, PhD, has decades-long experience in the telecommunications and Internet industries and has worked with both service providers and equipment vendors. He holds multiple industry-recognized certifications, such as JNCIE-SP, Emeritus. In his current role as a senior technical marketing engineer at Juniper Networks, Ridha has multiple responsibilities in projects involving data center technologies such as EVPN-VXLAN and, more recently, AI/ML Workloads.

Contents at a Glance

Contents

Introduction

My professional growth is built on the shoulders of tech and educational giants such as Jeff Doyle, Russ White, and Dinesh Dutt and their work. They have inspired generations, and just as their work inspired me, I hope this book inspires many others.

This book is a culmination of over a decade of technical learning and writing, working through customer escalations and designing, implementing, and troubleshooting small to large-scale enterprise and data center networks. And thus, this book is rooted in servant leadership and experiential learning. The goal of this book is not only to *show* but also to help you *learn* the finer details, the foundational knowledge that largely does not change as data center networks continue to evolve over time. More generally, the goal is to help you develop a mindset and a sound methodology behind building and troubleshooting data center networks.

To that end, each chapter is written with an unwavering focus on the "why." My approach to learning new technologies has always been to understand the history behind how they evolved and what were the driving factors. In this book, I have adapted that approach to *teaching* you new technologies. Outside of focusing on the configuration that is necessary to build data centers with Junos, each chapter aims to unpack what happens behind the scenes to give you a deeper understanding of this infrastructure, while also providing historical context, wherever necessary.

By the end of this book, you will have gained expert-level knowledge about the following topics:

- The Junos CLI and how to navigate it
- The history and evolution of data centers, moving from three-tier designs to a Clos architecture, necessitated by the predominance of east-west traffic resulting from the rise of server virtualization and a shift to a microservices architecture
- The history and evolution of VXLAN, moving from a flood-and-learn model to coupling it with BGP EVPN for control plane dissemination of MAC addresses, while also providing Layer 3 reachability
- EVPN route types 1 through 5
- Building small to large-scale data centers using VXLAN with BGP EVPN and different overlay models, based on customer need, such as bridged overlay, edge-routed bridging, routed overlay, or host-routed bridging
- Connecting multiple data centers using different interconnect options such as over-the-top DCI or Integrated Interconnect with IP and MPLS transports
- Using Juniper Apstra to orchestrate data centers built using user intent with continuous validation of intent
- Using a network emulation tool such as Containerlab to build and deploy virtual lab infrastructure

While this book is not written with the intent of helping you to pass a specific certification exam, it does act as an excellent supplemental source for studying to obtain the JNCIA-DC, JNCIS-DC, JNCIP-DC, and JNCIE-DC certifications.

How This Book Is Organized

Although this book is intended to be read cover to cover, each chapter stands on its own and can be read individually, depending on your need. The first four chapters are introductory chapters, providing the proper historical context behind data center design and evolution, while also introducing the Junos CLI and how to navigate and use it. These chapters cover the following topics:

- **Chapter 1, "Introducing the Juniper Ecosystem":** This chapter introduces the Juniper ecosystem with a focus on gaining familiarity with the Junos CLI by implementing common Layer 2 and Layer 3 features in a collapsed core design and using various **show** commands to validate user intent, including how to read and understand the MAC address table and various routing tables.
- **Chapter 2, "Overview of Data Center Architecture":** This chapter dives into the history and evolution of data centers, focused on the driving factors that influenced and led to these changes, moving from a traditional three-tier architecture to a Clos design.

- **Chapter 3, "BGP for the Data Center":** This chapter introduces how BGP is used for modern data centers built with a scale-out strategy using the Clos architecture.
- **Chapter 4, "VXLAN as a Network Virtualization Overlay":** This chapter introduces VXLAN as a network overlay, elevating network services into a logical layer on top of the physical infrastructure. It also provides historical context on how VXLAN evolved from using a flood-and-learn mechanism to using BGP EVPN as a control plane to disseminate MAC address information.

Chapters 5 through 11 form the core of the book. These provide the basic building blocks of designing and operating small to large-scale data centers. Chapter 5, especially, is the main building block of this book, introducing, and diving deeper into, core VXLAN with BGP EVPN functionality; it is foundational to every chapter that comes after it. These seven chapters cover the following topics:

- **Chapter 5, "Bridged Overlay in an EVPN VXLAN Fabric":** This chapter focuses on understanding, configuring, and validating a bridged overlay in an EVPN VXLAN fabric. It also provides a foundational understanding of how MAC addresses are learned in EVPN VXLAN fabrics and dives deeper into important aspects of such networks, such as how BUM traffic is replicated, EVPN multihoming, Route Targets, MAC mobility, loop detection, and Bidirectional Forwarding Detection.
- **Chapter 6, "MAC-VRFs":** This chapter introduces MAC-VRFs, a construct that provides Layer 2 multitenancy in EVPN VXLAN fabrics. This chapter also explores different EVPN service types such as VLAN-Based and VLAN-Aware.
- **Chapter 7, "Centrally Routed Bridging":** This chapter introduces the concept of integrated routed bridging and explores routing in EVPN VXLAN fabrics using a centrally routed bridging model.
- **Chapter 8, "Edge-Routed Bridging":** This chapter builds on the previous chapter, introducing the edge-routed bridging design, while exploring the asymmetric and symmetric routing models.
- **Chapter 9, "Routed Overlay and Host-Routed Bridging":** This chapter introduces the routed overlay and host-routed bridging designs, commonly used in infrastructures with cloud-native applications, with no requirement of Layer 2 overlays.
- **Chapter 10, "DHCP in EVPN VXLAN Fabrics":** This chapter introduces the challenges with DHCP in such routed fabrics, diving deeper into DHCP functionality in both bridged overlay and edge-routed bridging designs, while also exploring EVPN VXLAN network designs with a dedicated services VRF where the DHCP server is located.
- **Chapter 11, "Data Center Interconnect":** This chapter introduces how two or more data centers can be connected using the over-the-top DCI or Integrated Interconnect DCI options with IP or MPLS transports.

Chapters 12 through 15 introduce Juniper Apstra, an intent-based networking system, and dive deeper into how data centers can be deployed using Apstra. These chapters cover the following topics:

- **Chapter 12, "Building Data Centers with Juniper Apstra, Part I—Apstra Foundation":** This chapter provides a first look at Juniper Apstra and introduces the building blocks used in designing data centers with Apstra, demonstrating how these building blocks are used to build and deploy a bridged overlay EVPN VXLAN fabric.
- **Chapter 13, "Building Data Centers with Juniper Apstra, Part II—Advanced Apstra Deployments":** This chapter builds on the previous chapter, demonstrating how an edge-routed bridging design is built using Juniper Apstra. Various DCI options such as over-the-top DCI and Integrated Interconnect are also explored in detail in this chapter.
- **Chapter 14, "Building Virtual Fabrics with vJunos, Containerlab, and Juniper Apstra":** This chapter introduces the need for virtual network infrastructure and how to build it using Containerlab, enabling organizations to build digital twins for network validation and pre-change and post-change testing, usually integrated in a CI/CD pipeline.
- **Chapter 15, "Large-Scale Fabrics, Inter-VRF Routing, and Security Policies in Apstra":** The closing chapter of this book introduces and demonstrates how to build 5-stage Clos networks and the use of policies in Apstra to secure communication in EVPN VXLAN fabrics. This chapter also explores inter-VRF design options in Apstra.

Chapter 3

BGP for the Data Center

As described in Chapter 2, “Overview of Data Center Architecture,” modern data centers are built with a *scale-out* strategy (rather than a *scale-up* strategy), with predominantly east-west traffic as opposed to the north-south traffic in the traditional three-tier architecture. This shift in strategy was prompted by many factors, including the rise of server virtualization, deployment of high-density server clusters (requiring inter-server communication), new technologies facilitating virtual machine migrations, a shift toward cloud-native applications and workloads, and, more recently, deployment of GPU clusters for artificial intelligence.

In line with this shift in strategy, data center topologies have evolved from a three-tier architecture to a 3-stage Clos architecture (and 5-stage Clos fabrics for large-scale data centers), with the need to eliminate protocols such as Spanning Tree, which made the infrastructure difficult (and more expensive) to operate and maintain due to its inherent nature of blocking redundant paths. Thus, a routing protocol was needed to convert the network natively into Layer 3, with ECMP for traffic forwarding across all available equal cost links. Operational expenditure (OPEX) considerations are equally important as well, since OPEX greatly exceeds capital expenditure (CAPEX) in most IT budgets—the goal should be using a simpler control plane, attempting to reduce control plane interaction as much as possible, and minimizing network downtime due to complex protocols.

In the past, BGP has been used primarily in service provider networks, to provide reachability between autonomous systems globally. BGP was (and still is) the protocol of the Internet, for inter-domain routing. BGP, being a path vector protocol, relies on routing based on policy (with the autonomous system number [ASN] usually acting as a tie-breaker), compared to interior gateway protocols such as Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS), which use path selection based on a shortest path first logic.

RFC 7938, “Use of BGP for Routing in Large-Scale Data Centers,” provides merit to using BGP with a routed design for modern data centers with a 3-stage or 5-stage Clos architecture. For VXLAN fabrics, external BGP (eBGP) can be used for both the underlay and the overlay. This chapter provides a design and implementation perspective of how BGP is adapted for the data center, specifically with eBGP for the underlay, offering the following features for large-scale deployments:

- It enables a simpler implementation, relying on TCP for underlying transport and to establish adjacency between BGP speakers.
- Although BGP is assumed to be slower to converge, with minimal design changes and well-known ASN schemes, such problems are nonexistent.
- Implementing eBGP for the underlay (for the IPv4 or IPv6 address family) and eBGP for the overlay (for the EVPN address family) using BGP groups in Junos provides a clear, vertical separation of the underlay and the overlay.
- Using BGP for both the underlay and overlay provides a simpler operational and maintenance experience. Additionally, eBGP is generally considered easier to deploy and troubleshoot, with internal BGP (iBGP) considered to be more complicated with its need for route reflectors (or confederations) and its best path selection.

- Implementing auto-discovery of BGP neighbors using link-local IPv6 addressing and leveraging RFC 8950 (which obsoletes RFC 5549) to transport IPv4 Network Layer Reachability Information (NLRI) over an IPv6 peering for the underlay enables plug-and-play behavior for any new leafs and spines.

BGP Path Hunting and ASN Scheme for Data Centers

Every BGP-speaking system requires an ASN to be assigned to exchange network reachability information with other BGP-speaking systems. An iBGP peering is defined as two BGP speakers with the same ASN peering to each other; an eBGP peering is defined as two BGP speakers with different ASNs peering to each other. For the Internet, publicly owned and assigned ASNs are used (allocated by the *Internet Assigned Numbers Authority*, or *IANA*), but this is dangerous for private data centers. One of the most common outages on the Internet is caused by ASN hijacking, in which an organization advertises routes from an ASN that is publicly owned by a different organization or service provider.

For this reason, IANA provides a list of 16-bit and 32-bit private ASNs that organizations can use. The 16-bit private ASNs range from 65412 to 65534, giving only 1023 available ASNs for use. To overcome this limitation, IANA offers 32-bit private ASNs for use as well, providing a much larger range, from 4200000000 to 4294967294. It is imperative that organizations building their own private data centers use ASNs from these private ranges for internal peering.

BGP is designed to route between autonomous systems, where the destination IP prefix is chosen based on the shortest number of AS hops (assuming no policy modification). These AS hops are tracked as part of a BGP attribute called AS_PATH.

In a densely interconnected topology such as a 3-stage Clos network, BGP can suffer from a problem known as *path hunting*. Path hunting occurs when BGP, on losing a route, *hunts* for reachability to the destination via all other available paths, not knowing whether the route still exists in the network or not.

Consider the 3-stage Clos network shown in Figure 3-1, with every node assigned a unique ASN from the 16-bit private ASN range.

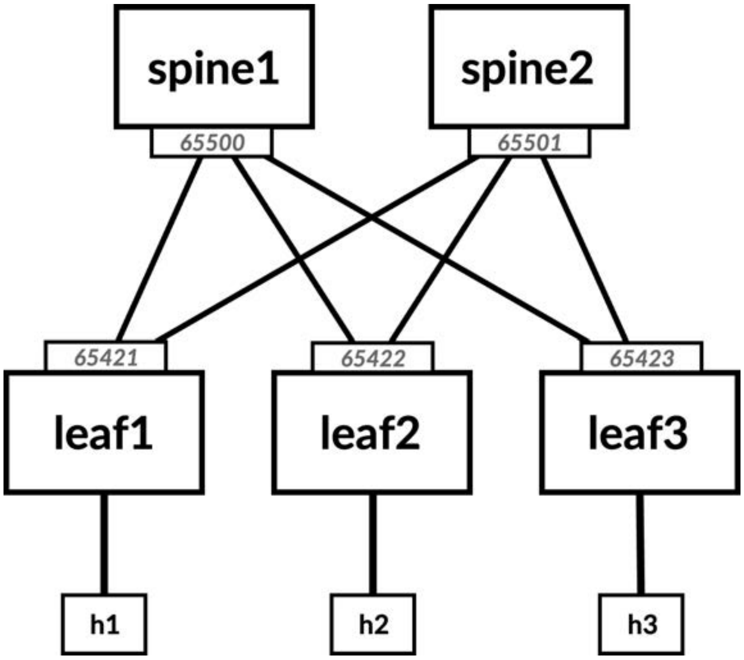


Figure 3-1 *Three-stage Clos network with unique ASNs per fabric node*

In this topology, leaf1 advertises a subnet x/y to spine1, as shown in Figure 3-2. This route is learned on spine1 with an AS_PATH attribute of [65421]. At the same time, the route is also advertised to spine2, and both spines advertise the route to leaf2 and leaf3.

BGP, by default, only advertises the best route to its neighbors. When leaf2 and leaf3 receive this route from both spine1 and spine2, they must elect one path as the best path. With no policy modification, the best path is chosen based on the shortest AS_PATH attribute, but in this case, the AS_PATH length is the same because the route received from spine1 will have an AS_PATH of [65500 65421] and the route received from spine2 will have an AS_PATH of [65501 65421]. Eventually, this tie-breaker is broken by selecting the oldest path. Assuming the elected best path is via spine2 (since it is the oldest path), leaf2 and leaf3 advertise this route to their eBGP peer list, which, in this case, consists only of spine1 (the route cannot be advertised back to spine2 because it originally sent the route that was elected as the best route).

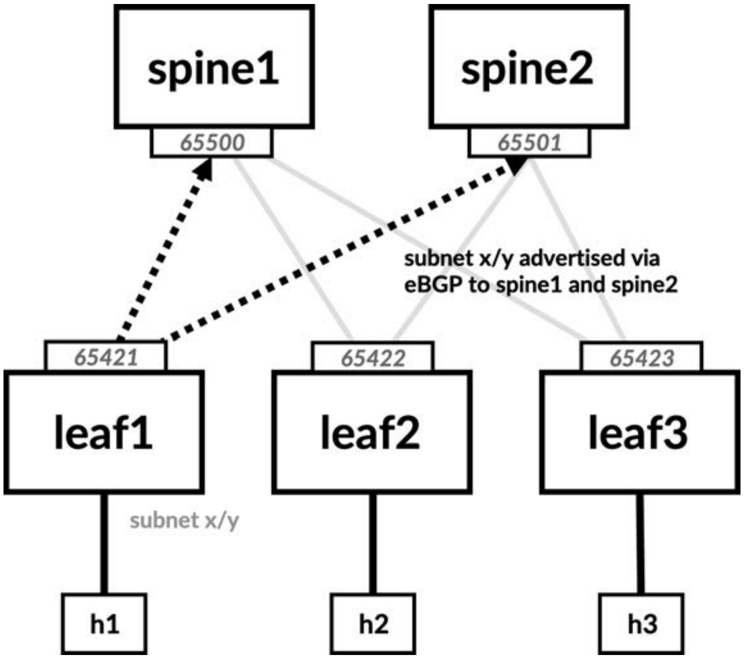
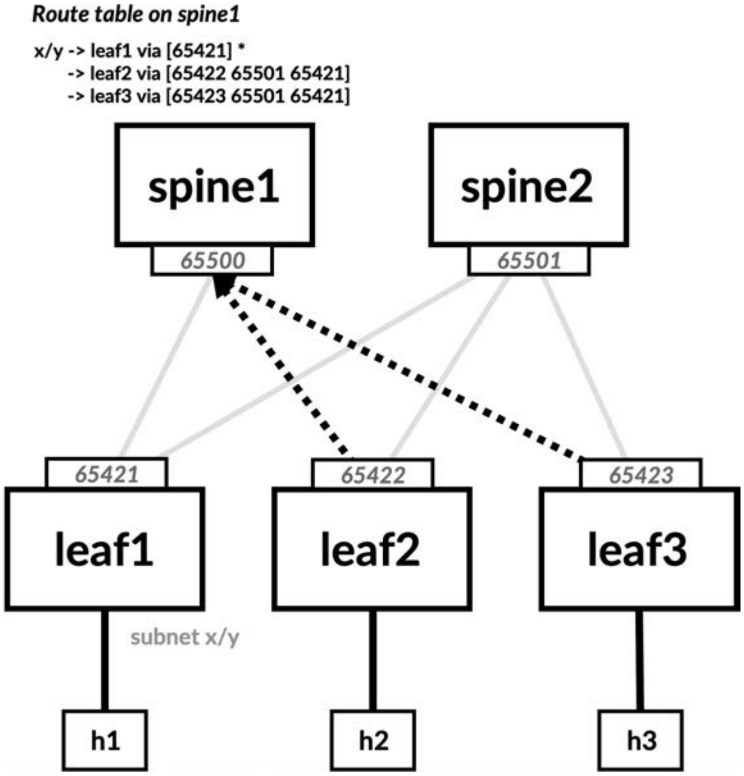


Figure 3-2 *Subnet x/y advertised to spine1 and spine2 by leaf1*

Thus, spine1 receives this route back from leaf2 and leaf3. At this point, spine1 has multiple paths available to reach subnet x/y advertised by leaf1; however, only the direct path (via leaf1) is selected as the best path, since it has the shortest AS_PATH length (again, assuming there are no policy modifications), as shown in Figure 3-3.



Route table on spine1

```
x/y -> leaf1 via [65421] *
    -> leaf2 via [65422 65501 65421]
    -> leaf3 via [65423 65501 65421]
```

Figure 3-3 *Routing table on spine1 showing all available paths for subnet x/y*

When spine1 loses its best path to subnet x/y, which is via leaf1 (leaf1 goes down or withdraws the route), it hunts for an alternate best path from all available paths. At the same time, spine1 also sends a BGP withdraw to its neighbors, informing them of the lost route via leaf1 for subnet x/y. Eventually, once all withdraws have converged and the subnet has been fully purged from the network, spine1 has no available paths for it, and the route is removed from its routing table.

While this path-hunting behavior might appear to be a minor problem, it becomes increasingly problematic as the fabric size increases with more leafs, creating many alternate paths to hunt through. Thus, to avoid this problem, and to speed up BGP convergence, either of the following two methodologies can be followed, with the same end goal of ensuring that the spines do not learn alternate, suboptimal routes reflected from other leafs:

- Use an ASN scheme, leveraging eBGP's built-in loop-prevention mechanism of dropping updates that include its own ASN in its AS_PATH list. This is the default BGP behavior, and you do need to configure any additional policies for this.
- Use routing policies to prevent spines from accepting routes that were originally advertised by any other spine.

This ASN scheme is represented in Figure 3-4.

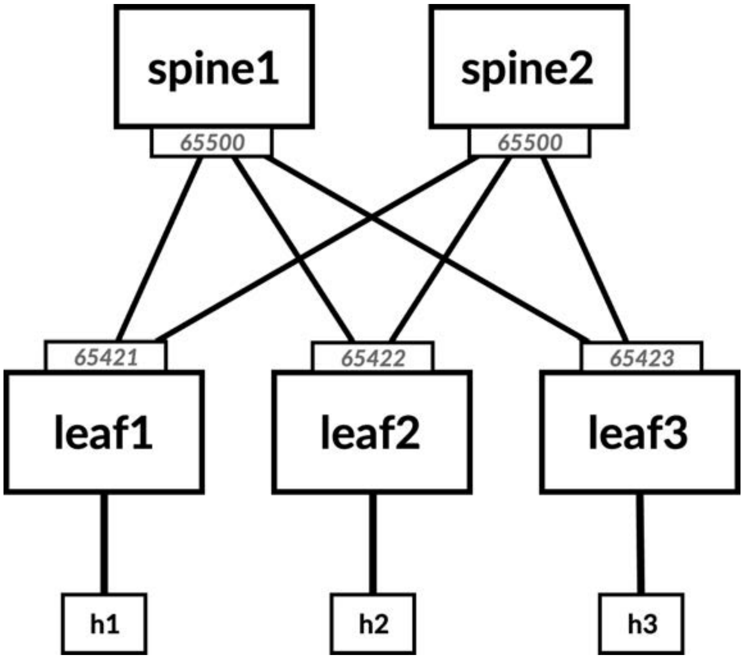


Figure 3-4 *BGP ASN scheme for a 3-stage Clos fabric to avoid path hunting with same ASN on all spines*

For a 5-stage Clos fabric, the ASN scheme mandates that all spines within a pod share the same ASN, but spines across pods have unique ASNs. Additionally, all leafs in each pod are assigned a unique ASN, while all superspines share the same ASN. This ASN scheme is represented in Figure 3-5.

Thus, for a 3-stage or 5-stage Clos fabric, with the ASN schemes shown in Figures 3-4 and 3-5, BGP path hunting is natively prevented.

The second methodology uses an ASN scheme in which all fabric nodes use a unique ASN, and routing policies are used to control how routes are advertised back to the spines to prevent BGP path hunting. In this case, as the spines advertise routes to the leafs, they are tagged with a BGP community using an export policy. On the leafs, an export policy is used to prevent the advertisement of routes with this BGP community from being sent back to the spines, thus preventing the existence of route state on the spines that can lead to path hunting. This is shown in Figure 3-6.

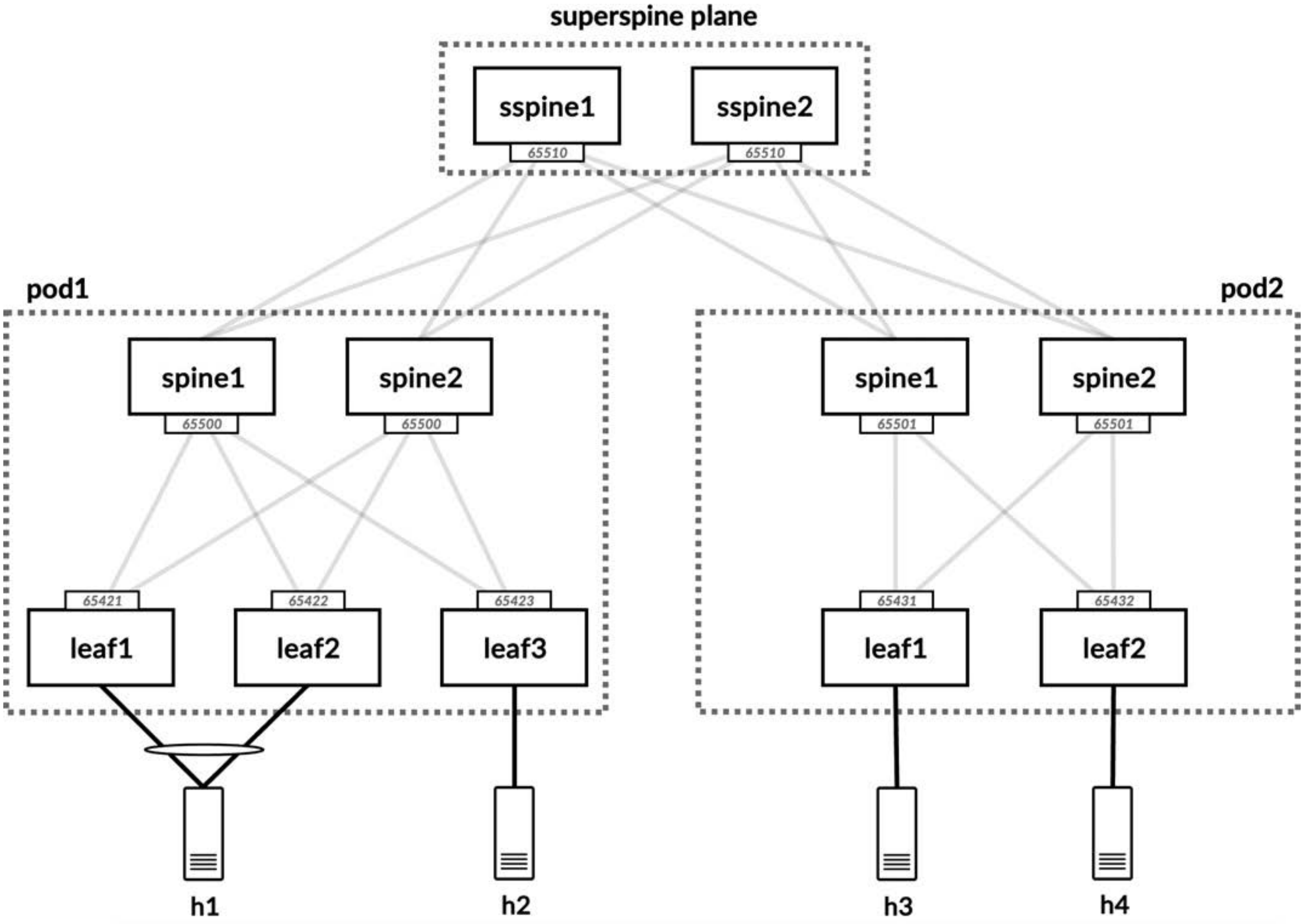


Figure 3-5 *BGP ASN scheme for a 5-stage fabric to avoid path hunting*

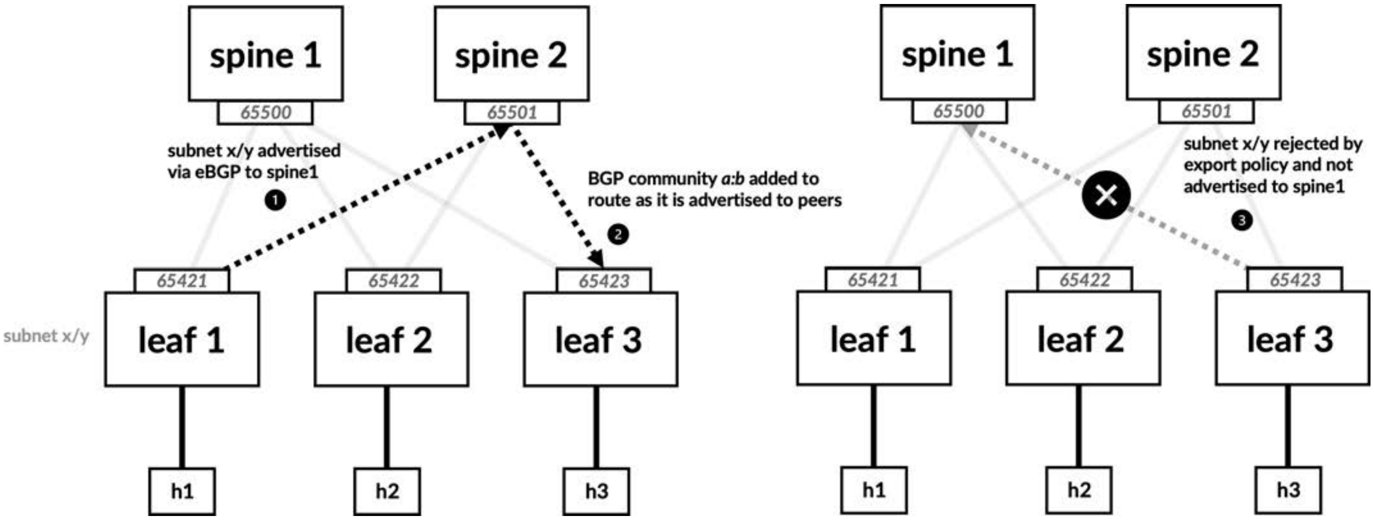


Figure 3-6 *Routing policy logic to prevent path hunting*

This implementation, while more complex and requiring additional operational overhead in the form of policy configuration, is necessary in certain designs where external devices are connected to the fabric for inter-VRF routing. Consider the topology shown in Figure 3-7, where the same ASN is used for both spines and a firewall is connected to leaf3 for inter-VRF routing.

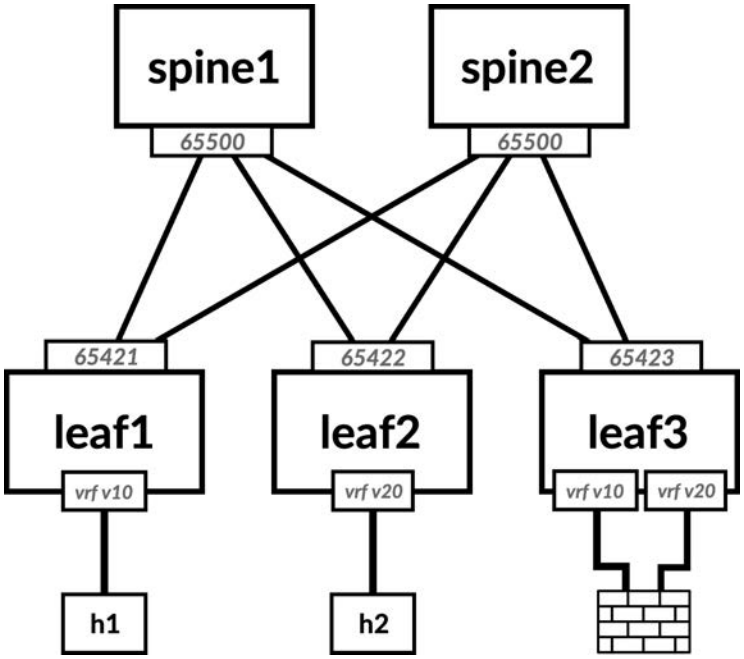


Figure 3-7 *Firewall connected to fabric leaf for inter-VRF routing*

In Figure 3-7, leaf1 is configured with an IP VRF *v10*, which includes an IPv4 subnet 172.16.10.0/24, and leaf2 is configured with an IP VRF *v20*, which includes an IPv4 subnet 172.16.20.0/24. The firewall has a BGP peering to leaf3 over both these IP VRFs to leak routes from one VRF to another.

The IPv4 subnet 172.16.10.0/24 is advertised by leaf1 toward leaf3, and eventually to the firewall, with an AS_PATH list of [65423 65500 65421], as shown in Figure 3-8.

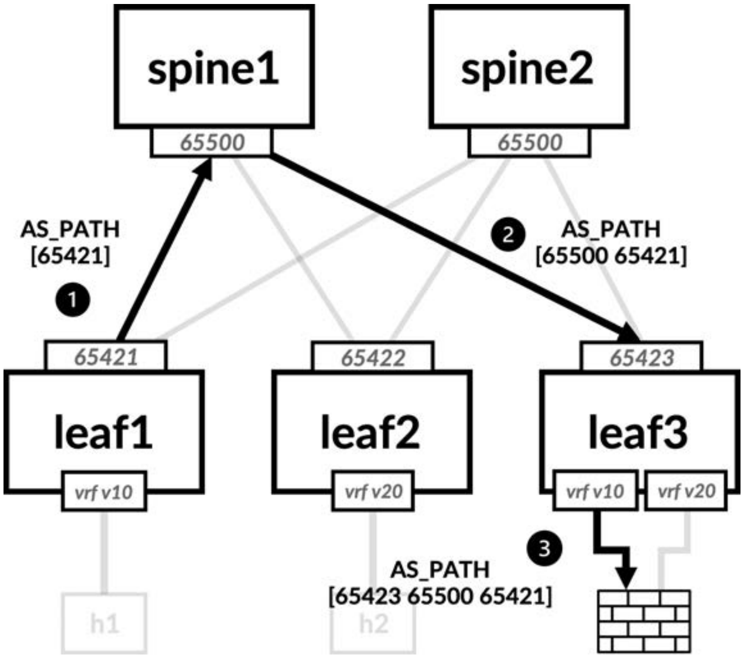


Figure 3-8 *AS_PATH attribute as a prefix, originated by leaf1, is advertised toward firewall*

The firewall "leaks" this route into IP VRF *v20* by advertising it to the VRF-specific BGP neighbor on leaf3. Thus, leaf3 receives this in IP VRF *v20* and advertises it to the rest of the fabric via the spines. However, when the spines receive this BGP update, they drop it because their local ASN is present in the AS_PATH list and BGP loop prevention rules indicate that such an update must be dropped. This is shown in Example 3-1, with BGP debugs on spine1.

Example 3-1 *Spines dropping BGP update due to AS loop prevention rules*

```
Jan 14 17:34:26.497233 BGP RECV 192.0.2.13+179 -> 192.0.2.101+61507
Jan 14 17:34:26.497273 BGP RECV message type 2 (Update) length 128
Jan 14 17:34:26.497369 BGP RECV Update PDU length 128
Jan 14 17:34:26.497452 BGP RECV flags 0x40 code Origin(1): IGP
Jan 14 17:34:26.497517 BGP RECV flags 0x40 code ASPath(2) length 22: 65423 65510 65423 65500 65421
Jan 14 17:34:26.497550 BGP RECV flags 0xc0 code Extended Communities(16): 2:502:502 encapsulation:vxlan(0x8) router-
mac:2c:6b:f5:75:70:f0
Jan 14 17:34:26.497561 BGP RECV flags 0x90 code MP_reach(14): AFI/SAFI 25/70
Jan 14 17:34:26.497577 BGP RECV nhop 192.0.2.13 len 4
Jan 14 17:34:26.497650 BGP RECV 5:192.0.2.14:502::0::172.16.10.0::24/248 (label field value 0x2906 [label 656, VNID
10502]) (esi 00:00:00:00:00:00:00:00:00:00)
Jan 14 17:34:26.497661 End-of-Attributes
Jan 14 17:34:26.497910 As loop detected. Rejecting update

*snip*
```

Figure 3-9 shows a visual representation of the same behavior.

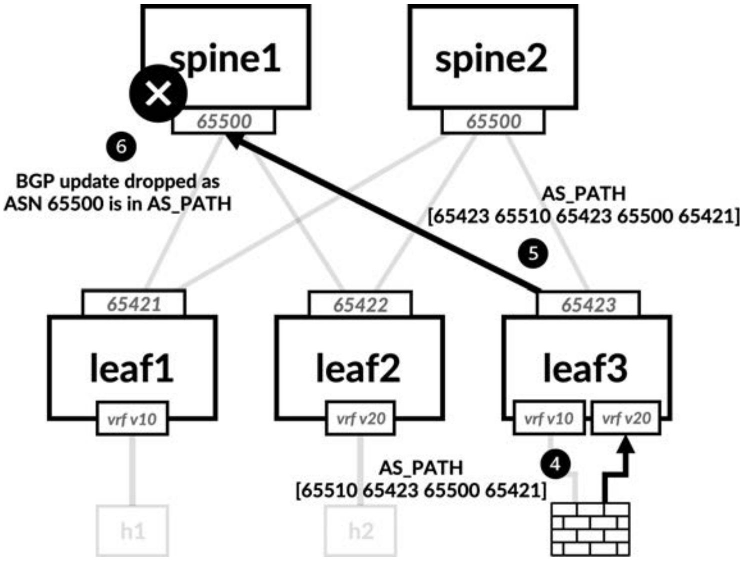


Figure 3-9 *BGP update dropped on spine1 due to local ASN 65500 in AS_PATH*

These problems can be circumvented by allowing the same ASN to be present in the AS_PATH attribute using several configuration options in Junos or by using an ASN scheme where each spine is assigned a unique ASN. Intent-based networking systems such as Juniper Apstra take away the complexity of implementing such an ASN scheme by automating and orchestrating the configuration of necessary policies to prevent path hunting (since that is the prevailing problem when each spine is assigned a unique ASN), with no requirement of operator intervention, while also facilitating designs as shown in Figure 3-7.

Implementing BGP for the Underlay

This section provides implementation specifics for building an eBGP underlay for an IP fabric or a VXLAN fabric using network devices running Junos. A unique ASN per fabric node design is used to demonstrate how spines can have suboptimal paths that can lead to path hunting, since the implementation of using the same ASNs on all spines in a 3-stage Clos network is straightforward and requires no demonstration. Then, routing policies are implemented to prevent path hunting. The implementation is based on the topology shown earlier in Figure 3-1.

In this network, for the underlay, each fabric-facing interface is configured as a point-to-point Layer 3 interface, as shown in Example 3-2 from the perspective of leaf1.

Example 3-2 *Point-to-point Layer 3 interface configuration on leaf1 for fabric-facing interfaces*

```
admin@leaf1# show interfaces ge-0/0/0
description "To spine1";
mtu 9100;
unit 0 {
    family inet {
        address 198.51.100.0/31;
    }
}

admin@leaf1# show interfaces ge-0/0/1
description "To spine2";
mtu 9100;
unit 0 {
    family inet {
        address 198.51.100.2/31;
    }
}
```

The goal of the underlay is to advertise the loopbacks of the *VXLAN Tunnel Endpoints (VTEPs)*, since these loopbacks are used to build end-to-end VXLAN tunnels. Thus, on each VTEP, which are the fabric leafs in this case, a loopback interface is configured, as shown on leaf1 in Example 3-3.

Example 3-3 *Loopback interface on leaf1*

```
admin@leaf1# show interfaces lo0
unit 0 {
    family inet {
        address 192.0.2.11/32;
    }
}
```

The underlay eBGP peering is between these point-to-point interfaces. Since a leaf's loopback address is sent toward other leafs via multiple spines, each leaf is expected to install multiple, equal cost paths to every other leaf's loopback address. In Junos, to enable ECMP routing, both the protocol (software) and the hardware need to be explicitly enabled to support it. In the case of BGP, this is enabled using the **multipath** knob (with the **multiple-as** configuration option if the routes received have the same AS_PATH length but different ASNs in the list). A subset of the eBGP configuration, for the underlay, is shown from the perspective of both spines and leaf1 in Example 3-4.

Example 3-4 *BGP configuration on spine1, spine2, and leaf1*

```
admin@spine1# show protocols bgp
group underlay {
    type external;
    family inet {
        unicast;
    }
    neighbor 198.51.100.0 {
        peer-as 65421;
    }
```

```
    neighbor 198.51.100.4 {
        peer-as 65422;
    }
    neighbor 198.51.100.8 {
        peer-as 65423;
    }
}

admin@spine2# show protocols bgp
group underlay {
    type external;
    family inet {
        unicast;
    }
    neighbor 198.51.100.2 {
        peer-as 65421;
    }
    neighbor 198.51.100.6 {
        peer-as 65422;
    }
    neighbor 198.51.100.10 {
        peer-as 65423;
    }
}

admin@leaf1# show protocols bgp
group underlay {
    type external;
    family inet {
        unicast;
    }
    export allow-loopback;
    multipath {
        multiple-as;
    }
    neighbor 198.51.100.1 {
        peer-as 65500;
    }
    neighbor 198.51.100.3 {
        peer-as 65501;
    }
}
```

Every leaf is advertising its loopback address via an export policy attached to the BGP group for the underlay, as shown in Example 3-4. The configuration of this policy is shown in Example 3-5, which enables the advertisement of direct routes in the 192.0.2.0/24 range to its eBGP peers.

Example 3-5 *Policy to advertise loopbacks shown on leaf1*

```
admin@leaf1# show policy-options policy-statement allow-loopback
term loopback {
    from {
        protocol direct;
        route-filter 192.0.2.0/24 orlonger;
    }
    then accept;
}
term discard {
    then reject;
}
```

> **Note** It is important to note that each routing protocol is associated with a default routing policy in Junos. For BGP, active BGP routes are readvertised to BGP speakers without the need of an export policy, while following protocol-specific rules, such as those for iBGP neighbors, which is why there is no need for an explicit export policy on the spines to advertise received routes from a leaf to all other leafs.

With the other leafs configured in the same way, the spines can successfully form an eBGP peering with each leaf, as shown in Example 3-6.

Example 3-6 *eBGP peering on spine1 and spine2 with all leafs*

```
admin@spine1> show bgp summary

Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 1 Peers: 3 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                       3          3          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Received/Accepted/Damped...
198.51.100.0          65421        191        189       0       0     1:24:41 Establ
  inet.0: 1/1/1/0
198.51.100.4          65422        184        182       0       0     1:21:12 Establ
  inet.0: 1/1/1/0
198.51.100.8          65423        180        179       0       0     1:19:35 Establ
  inet.0: 1/1/1/0

admin@spine2> show bgp summary

Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 1 Peers: 3 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                       3          3          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Received/Accepted/Damped...
```

```
198.51.100.2          65421        194        191       0       0     1:25:52 Establ
  inet.0: 1/1/1/0
198.51.100.6          65422        183        181       0       0     1:20:57 Establ
  inet.0: 1/1/1/0
198.51.100.10         65423        180        179       0       0     1:19:21 Establ
  inet.0: 1/1/1/0
```

With the policy configured as shown in Example 3-5, and the BGP peering between the leafs and the spines in an *Established* state, the loopback address of each leaf should be learned on every other leaf in the fabric.

Consider leaf1 now, to understand how equal cost paths for another leaf's loopback address are installed. For the loopback address of leaf2, advertised by both spine1 and spine2 to leaf1, two routes are received on leaf1. Since BGP is configured with **multipath**, both routes are installed as equal cost routes in software, as shown in Example 3-7.

Example 3-7 *Equal cost routes to leaf2's loopback on leaf1*

```
admin@leaf1> show route table inet.0 192.0.2.12

inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
+ = Active Route, - = Last Active, * = Both

192.0.2.12/32      *[BGP/170] 02:10:44, localpref 100, from 198.51.100.1
                      AS path: 65500 65422 I, validation-state: unverified
                       to 198.51.100.1 via ge-0/0/0.0
                    >  to 198.51.100.3 via ge-0/0/1.0
                    [BGP/170] 02:10:44, localpref 100
                      AS path: 65501 65422 I, validation-state: unverified
                    >  to 198.51.100.3 via ge-0/0/1.0
```

A validation-state of *unverified*, as shown in Example 3-7, implies that the BGP route validation feature has not been configured (this is a feature to validate the origin and the path of a BGP route, to ensure that it is legitimate), and the route has been accepted but it was not validated.

These equal cost routes must also be installed in hardware. This is achieved by configuring the Packet Forwarding Engine (PFE) to install equal cost routes, and in turn, program the hardware, by applying an export policy under the **routing-options** hierarchy, as shown in Example 3-8. The policy itself simply enables per-flow load balancing. This example also demonstrates how the forwarding table, on the Routing Engine, can be viewed for a specific destination IP prefix, using the **show route forwarding-table destination** [*ip-address*] **table** [*table-name*] operational mode command.

Example 3-8 *Equal cost routes in PFE of leaf1 with a policy for load-balancing per flow*

```
admin@leaf1# show routing-options forwarding-table
export ecmp;

admin@leaf1# show policy-options policy-statement ecmp
then {
    load-balance per-flow;
}

admin@leaf1> show route forwarding-table destination 192.0.2.12/32 table default
Routing table: default.inet
Internet:
Destination        Type RtRef Next hop           Type Index    NhRef Netif
```

```
192.0.2.12/32      user     0                  ulst  1048574     3
                            198.51.100.1       ucst      583     4 ge-0/0/0.0
                            198.51.100.3       ucst      582     4 ge-0/0/1.0
```

While the control plane and the route installation in both software and hardware are as expected on the leafs, the spines paint a different picture. If the loopback address of the leafs, advertised by spine1 to other leafs, is chosen as the best route, spine2 will receive and store all suboptimal paths in its routing table. Again, considering leaf1's loopback address as an example here, spine2 has three paths for this route, as shown in Example 3-9.

Example 3-9 *Multiple paths for leaf1's loopback address on spine2*

```
admin@spine2> show route table inet.0 192.0.2.11/32

inet.0: 10 destinations, 16 routes (10 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
+ = Active Route, - = Last Active, * = Both

192.0.2.11/32      *[BGP/170] 15:05:38, localpref 100
                      AS path: 65421 I, validation-state: unverified
                    >  to 198.51.100.2 via ge-0/0/0.0
                    [BGP/170] 00:02:39, localpref 100
                      AS path: 65422 65500 65421 I, validation-state: unverified
                    >  to 198.51.100.6 via ge-0/0/1.0
                    [BGP/170] 00:01:02, localpref 100
                      AS path: 65423 65500 65421 I, validation-state: unverified
                    >  to 198.51.100.10 via ge-0/0/2.0
```

This includes the direct path via leaf1, an indirect path via leaf2, and another indirect path via leaf3. Thus, in this case, if spine2 loses the direct path via leaf1, it will start path hunting through the other suboptimal paths, until the network fully converges with all withdraws processed on all fabric nodes. This problem can be addressed by applying an export policy on the spines that adds a BGP community to all advertised routes, and then using this community on the leafs to match and reject such routes from being advertised back to the spines.

In Junos, a routing policy controls the import of routes into the routing table and the export of routes from the routing table, to be advertised to neighbors. In general, a routing policy consists of terms, which include match conditions and associated actions. The routing policy on the spines is shown in Example 3-10 and includes the following two policy terms:

- **all-bgp:** Matches all BGP learned routes, accepts them, and adds a community value from the community name spine-to-leaf.
- **loopback:** Matches all direct routes in the IPv4 subnet 192.0.2.0/24. The **orlonger** configuration option matches any IPv4 address that is equal to or longer than the defined prefix length.

Example 3-10 *Policy to add a BGP community on the spines as they advertise routes to leafs*

```
admin@spine2# show policy-options policy-statement spine-to-leaf
term all-bgp {
    from protocol bgp;
    then {
        community add spine-to-leaf;
        accept;
    }
}
```

```
term loopback {
    from {
        protocol direct;
        route-filter 192.0.2.0/24 orlonger;
    }
    then {
        community add spine-to-leaf;
        accept;
    }
}

admin@spine2# show policy-options community spine-to-leaf
members 0:15;
```

Once the policy in Example 3-10 is applied as an export policy on the spines for the underlay BGP group, the leafs receive all BGP routes attached with a BGP community of value 0:15. This can be confirmed on leaf2, taking leaf1's loopback address into consideration, as shown in Example 3-11.

Example 3-11 *Leaf1's loopback address received with a BGP community of 0:15 on leaf2*

```
admin@leaf2> show route table inet.0 192.0.2.11/32 extensive

inet.0: 9 destinations, 12 routes (9 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
192.0.2.11/32 (2 entries, 1 announced)
TSI:
KRT in-kernel 192.0.2.11/32 -> {list:198.51.100.5, 198.51.100.7}
Page 0 idx 0, (group underlay type External) Type 1 val 0x85194a0 (adv_entry)
   Advertised metrics:
     Nexthop: 198.51.100.5
     AS path: [65422] 65500 65421 I
     Communities: 0:15
    Advertise: 00000002
Path 192.0.2.11
from 198.51.100.5
Vector len 4.  Val: 0
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 0
                Address: 0x7a46fac
                Next-hop reference count: 3, Next-hop session id: 0
                Kernel Table Id: 0
                Source: 198.51.100.5
                Next hop: 198.51.100.5 via ge-0/0/0.0
                Session Id: 0
                Next hop: 198.51.100.7 via ge-0/0/1.0, selected
                Session Id: 0
                State: <Active Ext>
                Local AS: 65422 Peer AS: 65500
                Age: 3:35
```

```
                Validation State: unverified
                Task: BGP_65500.198.51.100.5
                Announcement bits (3): 0-KRT 1-BGP_Multi_Path 2-BGP_RT_Background
                AS path: 65500 65421 I
                Communities: 0:15
                Accepted Multipath
                Localpref: 100
                Router ID: 192.0.2.101
                Thread: junos-main
         BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 577
                Address: 0x77c63f4
                Next-hop reference count: 5, Next-hop session id: 321
                Kernel Table Id: 0
                Source: 198.51.100.7
                Next hop: 198.51.100.7 via ge-0/0/1.0, selected
                Session Id: 321
                State: <Ext>
                Inactive reason: Active preferred
                Local AS: 65422 Peer AS: 65501
                Age: 5:30
                Validation State: unverified
                Task: BGP_65501.198.51.100.7
                AS path: 65501 65421 I
                Communities: 0:15
                Accepted MultipathContrib
                Localpref: 100
                Router ID: 192.0.2.102
                Thread: junos-main
```

On the leafs, it is now a simple matter of rejecting any route that has this community to stop it from being readvertised back to the spines. A new policy is created for this, and it is applied using an *and* operation to the existing policy that advertises the loopback address, as shown in Example 3-12 from the perspective of leaf1.

Example 3-12 *Policy on leaf1 to reject BGP routes with a community of 0:15*

```
admin@leaf1# show policy-options policy-statement leaf-to-spine
term reject-to-spine {
    from {
        protocol bgp;
        community spine-to-leaf;
    }
    then reject;
}
term accept-all {
    then accept;
}

admin@leaf1# show policy-options community spine-to-leaf
members 0:15;
```

```
admin@leaf1# show protocols bgp
group underlay {
    type external;
    family inet {
        unicast;
    }
    export ( leaf-to-spine && allow-loopback );
    multipath {
        multiple-as;
    }
    neighbor 198.51.100.1 {
        peer-as 65500;
    }
    neighbor 198.51.100.3 {
        peer-as 65501;
    }
}
```

With this policy applied on all the leafs, the spines will not learn any suboptimal paths to each of the leaf loopbacks. This is confirmed in Example 3-13, with each spine learning every leaf's loopback address via the direct path to the respective leaf.

Example 3-13 *Route to each leaf's loopback address on spine1 and spine2*

```
admin@spine1> show route table inet.0 192.0.2.11/32

inet.0: 10 destinations, 10 routes (10 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
+ = Active Route, - = Last Active, * = Both

192.0.2.11/32      *[BGP/170] 15:45:36, localpref 100
                      AS path: 65421 I, validation-state: unverified
                    >  to 198.51.100.0 via ge-0/0/0.0

admin@spine1> show route table inet.0 192.0.2.12/32

inet.0: 10 destinations, 10 routes (10 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
+ = Active Route, - = Last Active, * = Both

192.0.2.12/32      *[BGP/170] 15:42:09, localpref 100
                      AS path: 65422 I, validation-state: unverified
                    >  to 198.51.100.4 via ge-0/0/1.0

admin@spine1> show route table inet.0 192.0.2.13/32

inet.0: 10 destinations, 10 routes (10 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
+ = Active Route, - = Last Active, * = Both
```

```
192.0.2.13/32      *[BGP/170] 15:40:35, localpref 100
                      AS path: 65423 I, validation-state: unverified
                   >  to 198.51.100.8 via ge-0/0/2.0

admin@spine2> show route table inet.0 192.0.2.11/32

inet.0: 10 destinations, 10 routes (10 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
+ = Active Route, - = Last Active, * = Both

192.0.2.11/32      *[BGP/170] 15:47:10, localpref 100
                      AS path: 65421 I, validation-state: unverified
                   >  to 198.51.100.2 via ge-0/0/0.0

admin@spine2> show route table inet.0 192.0.2.12/32

inet.0: 10 destinations, 10 routes (10 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
+ = Active Route, - = Last Active, * = Both

192.0.2.12/32      *[BGP/170] 15:42:18, localpref 100
                      AS path: 65422 I, validation-state: unverified
                   >  to 198.51.100.6 via ge-0/0/1.0

admin@spine2> show route table inet.0 192.0.2.13/32

inet.0: 10 destinations, 10 routes (10 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
+ = Active Route, - = Last Active, * = Both

192.0.2.13/32      *[BGP/170] 15:40:45, localpref 100
                      AS path: 65423 I, validation-state: unverified
                   >  to 198.51.100.10 via ge-0/0/2.0
```

Junos also offers the operator a direct way to test the policy, which can be used to confirm that a leaf's locally owned loopback address is being advertised to the spines, and other loopback addresses learned via BGP are rejected. This uses the **test policy** operational mode command, as shown in Example 3-14, where only leaf1's loopback address (192.0.2.11/32) is accepted by the policy, while leaf2's and leaf3's loopback addresses, 192.0.2.12/32 and 192.0.2.13/32 respectively, are rejected by the policy.

Example 3-14 *Policy rejecting leaf2's and leaf3's loopback addresses from being advertised to the spines on leaf1*

```
admin@leaf1> test policy leaf-to-spine 192.0.2.11/32

inet.0: 9 destinations, 11 routes (9 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
+ = Active Route, - = Last Active, * = Both

192.0.2.11/32      *[Direct/0] 1d 04:38:27
                   >  via lo0.0
```

```
Policy leaf-to-spine: 1 prefix accepted, 0 prefix rejected

admin@leaf1> test policy leaf-to-spine 192.0.2.12/32

Policy leaf-to-spine: 0 prefix accepted, 1 prefix rejected

admin@leaf1> test policy leaf-to-spine 192.0.2.13/32

Policy leaf-to-spine: 0 prefix accepted, 1 prefix rejected
```

With this configuration in place, the fabric underlay is successfully built, with each leaf's loopback address reachable from every other leaf, as shown in Example 3-15, while also preventing any path-hunting issues on the spines by using appropriate routing policies.

Example 3-15 *Loopback reachability from leaf1*

```
admin@leaf1> ping 192.0.2.12 source 192.0.2.11
PING 192.0.2.12 (192.0.2.12): 56 data bytes
64 bytes from 192.0.2.12: icmp_seq=0 ttl=63 time=3.018 ms
64 bytes from 192.0.2.12: icmp_seq=1 ttl=63 time=2.697 ms
64 bytes from 192.0.2.12: icmp_seq=2 ttl=63 time=4.773 ms
64 bytes from 192.0.2.12: icmp_seq=3 ttl=63 time=3.470 ms
^C
--- 192.0.2.12 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.697/3.490/4.773/0.790 ms

admin@leaf1> ping 192.0.2.13 source 192.0.2.11
PING 192.0.2.13 (192.0.2.13): 56 data bytes
64 bytes from 192.0.2.13: icmp_seq=0 ttl=63 time=2.979 ms
64 bytes from 192.0.2.13: icmp_seq=1 ttl=63 time=2.814 ms
64 bytes from 192.0.2.13: icmp_seq=2 ttl=63 time=2.672 ms
64 bytes from 192.0.2.13: icmp_seq=3 ttl=63 time=2.379 ms
^C
--- 192.0.2.13 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.379/2.711/2.979/0.220 ms
```

Auto-Discovered BGP Neighbors

The previous section demonstrated how to build an eBGP-based fabric underlay using point-to-point Layer 3 interfaces. This requires extensive IP management and operational maintenance as the fabric grows. An alternate, more efficient approach is to use a BGP feature called *BGP auto-discovery* (also referred to as *BGP unnumbered*), which uses link-local IPv6 addressing to automatically peer with its discovered neighbor by leveraging IPv6 Neighbor Discovery (ND). This is very beneficial for several reasons:

- It eliminates the need for IP address management of the underlay and enables plug-and-play insertion of new fabric nodes.
- It allows for easier automation of the underlay of the fabric since every fabric interface is configured the same way, with no IP addressing required. BGP, unlike IGPs, is designed to peer with untrusted neighbors, and thus the default need to

specify a peer address, assign an ASN, and configure authentication for BGP peering. In a data center, which is largely a trusted environment, BGP is utilized more like an IGP, which makes automating it much easier, reducing any configuration complexity.

This section provides an implementation example of how to configure and deploy BGP auto-discovery, using packet captures for a deeper understanding of the same. The topology shown in Figure 3-10 is used to demonstrate this feature.

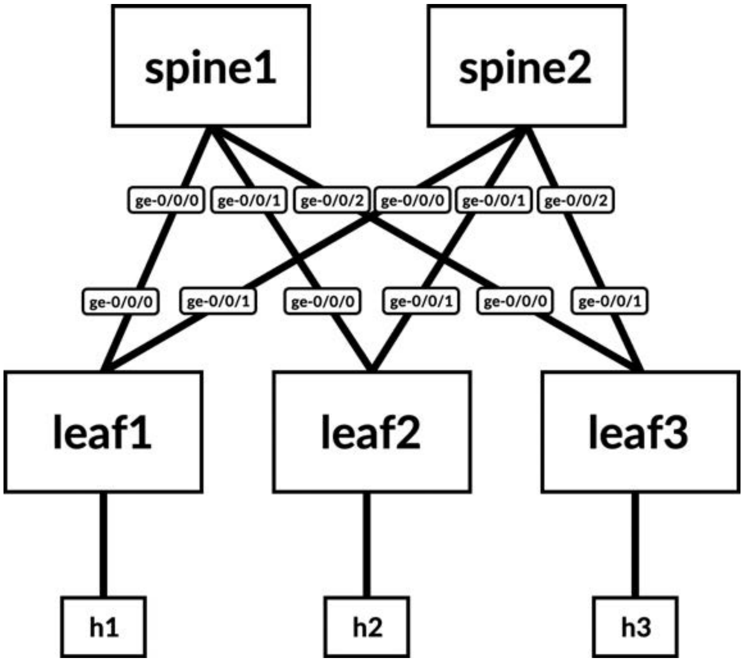


Figure 3-10 *Topology to implement BGP auto-discovered neighbors*

BGP auto-discovery relies on IPv6 Neighbor Discovery Protocol (NDP), which uses ICMPv6 messages to announce its link-local IPv6 address to its directly attached neighbors and learn the neighbors' link-local IPv6 addresses from inbound ICMPv6 messages, replacing the traditional IPv4 ARP process. More specifically, this is achieved using an ICMPv6 message type called *Router Advertisement (RA)*, which has an opcode of 134.

To enable BGP auto-discovery, the following steps must be done:

- Enable IPv6 on the fabric-facing point-to-point interfaces. The IPv4 family must be enabled as well if IPv4 traffic is expected on the interface. Even though the peering between neighbors uses IPv6, the interface can carry traffic for any address family. No IPv6 or IPv4 address is required to be configured on these interfaces.
- Enable **protocol router-advertisements** on the fabric-facing interfaces (the default RA interval is 15 seconds).
- Configure BGP to automatically discover peers using IPv6 ND by enabling the underlay group for the IPv6 unicast address family and using the **dynamic-neighbor** hierarchy to define neighbor discovery using IPv6 ND for the fabric-facing interfaces.
- Configure BGP for the IPv4 unicast address family, with the **extended-nexthop** configuration option. This allows IPv4 routes to be advertised via BGP with an IPv6 next-hop using a new BGP capability defined in RFC 8950 (which obsoletes RFC 5549) called the Extended Next Hop Encoding capability. This capability is exchanged in the BGP OPEN message.

The configuration of spine1 is shown in Example 3-16 as a reference. For the spines, since each leaf is in a different ASN, the **peer-as-list** configuration option is used to specify a list of allowed peer ASNs to which a BGP peering can be established. It is important that this peer ASN list be carefully curated, since a peering request from any other ASN (outside of this list) will be rejected.

Example 3-16 *BGP auto-discovery configuration on spine1*

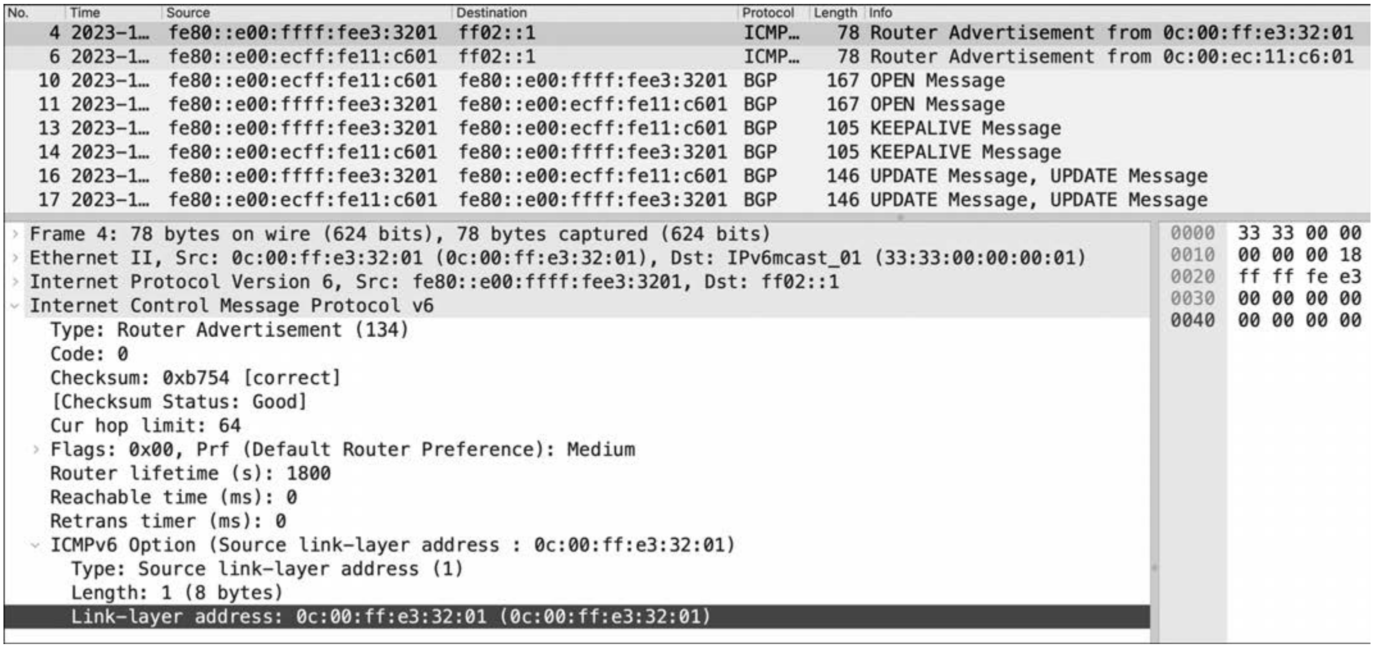
```
admin@spine1# show interfaces
ge-0/0/0 {
    unit 0 {
        family inet;
        family inet6;
    }
}
ge-0/0/1 {
    unit 0 {
        family inet;
        family inet6;
    }
}
ge-0/0/2 {
    unit 0 {
        family inet;
        family inet6;
    }
}

admin@spine1# show protocols router-advertisement
interface ge-0/0/0.0;
interface ge-0/0/1.0;
interface ge-0/0/2.0;

admin@spine1# show protocols bgp
group auto-underlay {
    family inet {
        unicast {
            extended-nexthop;
        }
    }
    family inet6 {
        unicast;
    }
    dynamic-neighbor underlay {
        peer-auto-discovery {
            family inet6 {
                ipv6-nd;
            }
            interface ge-0/0/0.0;
            interface ge-0/0/1.0;
            interface ge-0/0/2.0;
        }
    }
    peer-as-list leafs;
}
```

Once the respective fabric interfaces are enabled with IPv6 RA, the fabric nodes discover each other's link-local IPv6 addresses. For example, leaf1 has discovered spine1's and spine2's link-local IPv6 addresses (as well as the corresponding MAC addresses) over its directly attached interfaces, as shown in Example 3-17, using the **show ipv6 neighbors** operational mode command.

Example 3-17 *IPv6 neighbors discovered using RA on leaf1*

```
admin@leaf1> show ipv6 neighbors
IPv6 Address                        Linklayer Address  State      Exp   Rtr  Secure  Interface
fe80::e00:b3ff:fe09:1001             0c:00:b3:09:10:01  reachable  9     yes  no      ge-0/0/1.0
fe80::e00:ffff:fee3:3201             0c:00:ff:e3:32:01  reachable  14    yes  no      ge-0/0/0.0
Total entries: 2
```

This process of sending Router Advertisements can be seen in the packet capture shown in Figure 3-11, from the perspective of the link between leaf1 and spine1.



```
No.  Time      Source                    Destination               Protocol Length Info
   4 2023-1…  fe80::e00:ffff:fee3:3201  ff02::1                   ICMP…     78 Router Advertisement from 0c:00:ff:e3:32:01
   6 2023-1…  fe80::e00:ecff:fe11:c601  ff02::1                   ICMP…     78 Router Advertisement from 0c:00:ec:11:c6:01
  10 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  BGP      167 OPEN Message
  11 2023-1…  fe80::e00:ffff:fee3:3201  fe80::e00:ecff:fe11:c601  BGP      167 OPEN Message
  13 2023-1…  fe80::e00:ffff:fee3:3201  fe80::e00:ecff:fe11:c601  BGP      105 KEEPALIVE Message
  14 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  BGP      105 KEEPALIVE Message
  16 2023-1…  fe80::e00:ffff:fee3:3201  fe80::e00:ecff:fe11:c601  BGP      146 UPDATE Message, UPDATE Message
  17 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  BGP      146 UPDATE Message, UPDATE Message

> Frame 4: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
> Ethernet II, Src: 0c:00:ff:e3:32:01 (0c:00:ff:e3:32:01), Dst: IPv6mcast_01 (33:33:00:00:00:01)
> Internet Protocol Version 6, Src: fe80::e00:ffff:fee3:3201, Dst: ff02::1
v Internet Control Message Protocol v6
    Type: Router Advertisement (134)
    Code: 0
    Checksum: 0xb754 [correct]
    [Checksum Status: Good]
    Cur hop limit: 64
  > Flags: 0x00, Prf (Default Router Preference): Medium
    Router lifetime (s): 1800
    Reachable time (ms): 0
    Retrans timer (ms): 0
  v ICMPv6 Option (Source link-layer address : 0c:00:ff:e3:32:01)
      Type: Source link-layer address (1)
      Length: 1 (8 bytes)
      Link-layer address: 0c:00:ff:e3:32:01 (0c:00:ff:e3:32:01)

0000  33 33 00 00
0010  00 00 00 18
0020  ff ff fe e3
0030  00 00 00 00
0040  00 00 00 00
```

Figure 3-11 *Packet capture of ICMPv6 Router Advertisement*

Packet #4, highlighted in Figure 3-11, is an ICMPv6 Router Advertisement sent by spine1, while packet #5 is an ICMPv6 Router Advertisement sent by leaf1. Such packets are sent using the link-local IPv6 address as the source, destined to the well-known IPv6 multicast group of FF02::1. The link-local IPv6 address of leaf1's interface can be confirmed as shown in Example 3-18.

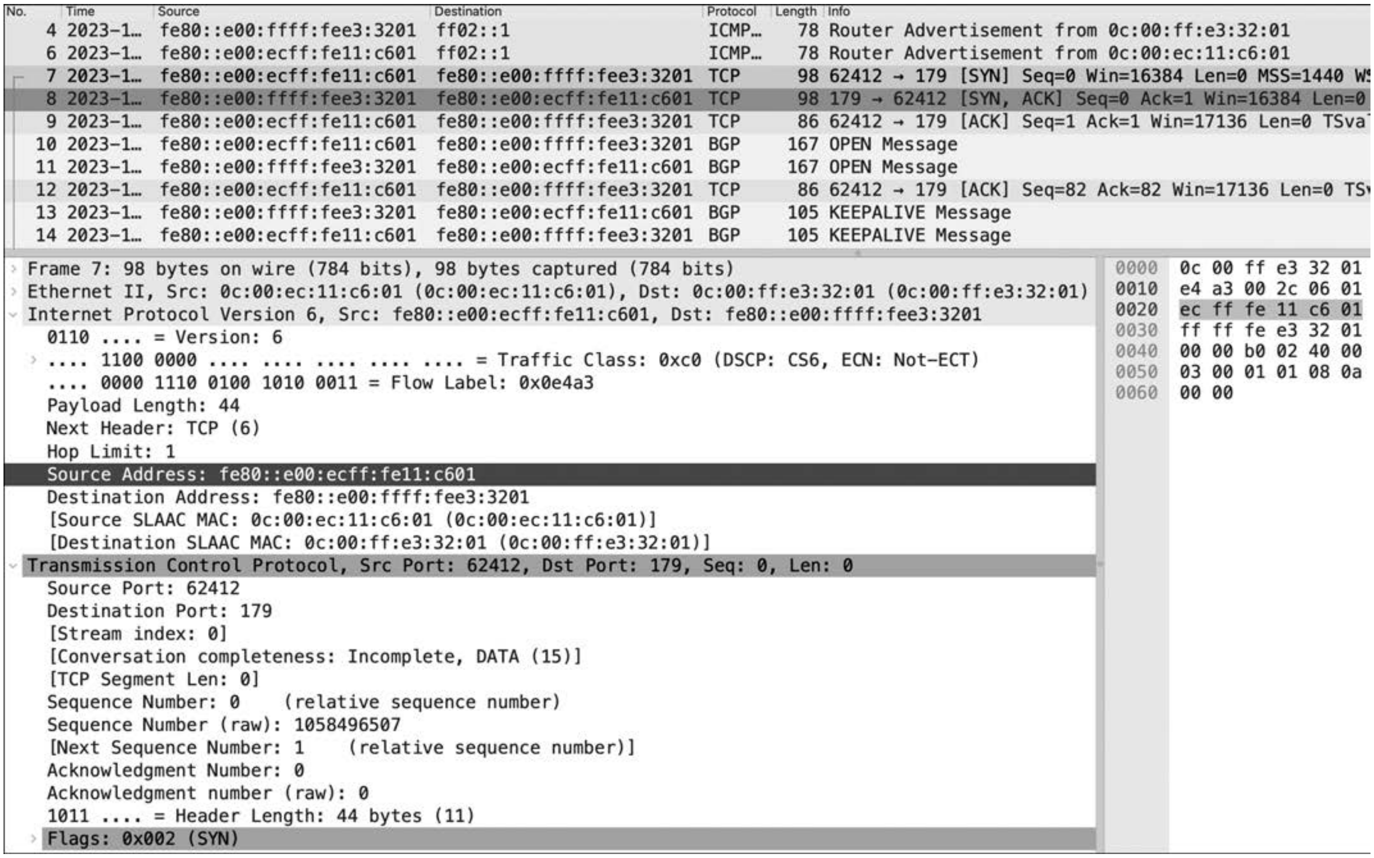
Example 3-18 *IPv6 link-local address assigned to ge-0/0/0.0 on leaf1*

```
admin@leaf1> show interfaces ge-0/0/0.0
  Logical interface ge-0/0/0.0 (Index 349) (SNMP ifIndex 540)
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Input packets : 847
    Output packets: 857
    Protocol inet, MTU: 1500
    Max nh cache: 100000, New hold nh limit: 100000, Curr nh cnt: 0, Curr new hold cnt: 0,
    NH drop cnt: 0
      Flags: Sendbcast-pkt-to-re, Is-Primary, 0x0
```

```
Protocol inet6, MTU: 1500
Max nh cache: 100000, New hold nh limit: 100000, Curr nh cnt: 1, Curr new hold cnt: 0,
NH drop cnt: 0
  Flags: Is-Primary, 0x0
  Addresses, Flags: Is-Preferred 0x800
    Destination: fe80::/64, Local: fe80::e00:ecff:fe11:c601
Protocol multiservice, MTU: Unlimited
  Flags: Is-Primary, 0x0
```

With the link-local IPv6 addresses discovered for a given link, a TCP session can be initiated to establish BGP peering between the fabric nodes. The entire communication is IPv6 only, including the initial TCP three-way handshake and all the BGP messages exchanged between the prospective neighbors, such as the BGP OPEN and the BGP UPDATE messages shown in Figure 3-11.

The entire handshake, as well as the instantiation of the BGP session, is shown in Figure 3-12 as a reference.



```
No.  Time       Source                     Destination                Protocol Length Info
   4 2023-1…  fe80::e00:ffff:fee3:3201  ff02::1                    ICMP…    78 Router Advertisement from 0c:00:ff:e3:32:01
   6 2023-1…  fe80::e00:ecff:fe11:c601  ff02::1                    ICMP…    78 Router Advertisement from 0c:00:ec:11:c6:01
   7 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  TCP      98 62412 → 179 [SYN] Seq=0 Win=16384 Len=0 MSS=1440 W
   8 2023-1…  fe80::e00:ffff:fee3:3201  fe80::e00:ecff:fe11:c601  TCP      98 179 → 62412 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0
   9 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  TCP      86 62412 → 179 [ACK] Seq=1 Ack=1 Win=17136 Len=0 TSva
  10 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  BGP     167 OPEN Message
  11 2023-1…  fe80::e00:ffff:fee3:3201  fe80::e00:ecff:fe11:c601  BGP     167 OPEN Message
  12 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  TCP      86 62412 → 179 [ACK] Seq=82 Ack=82 Win=17136 Len=0 TS
  13 2023-1…  fe80::e00:ffff:fee3:3201  fe80::e00:ecff:fe11:c601  BGP     105 KEEPALIVE Message
  14 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  BGP     105 KEEPALIVE Message
```

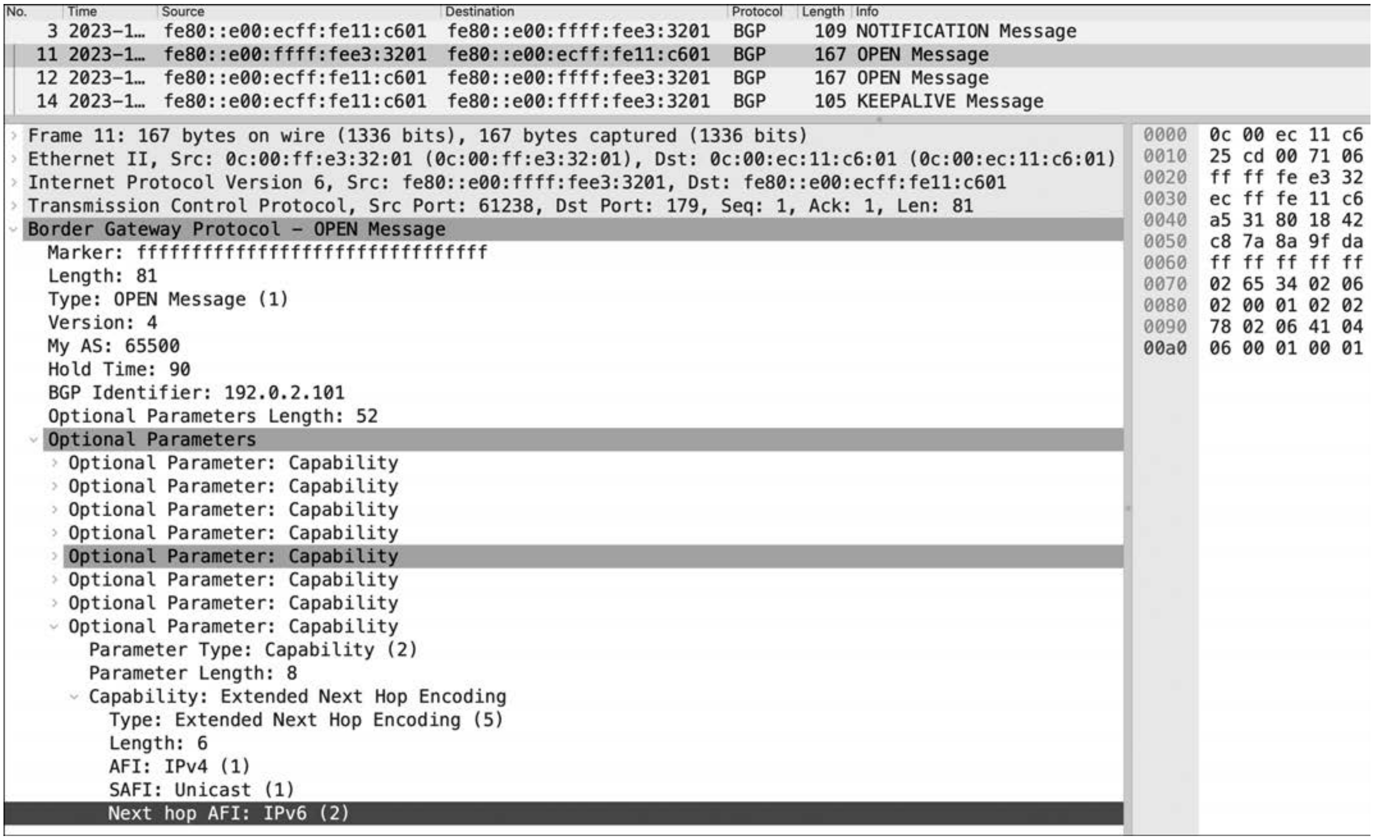
```
> Frame 7: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
> Ethernet II, Src: 0c:00:ec:11:c6:01 (0c:00:ec:11:c6:01), Dst: 0c:00:ff:e3:32:01 (0c:00:ff:e3:32:01)
v Internet Protocol Version 6, Src: fe80::e00:ecff:fe11:c601, Dst: fe80::e00:ffff:fee3:3201
    0110 .... = Version: 6
  > .... 1100 0000 .... .... .... .... .... = Traffic Class: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    .... 0000 1110 0100 1010 0011 = Flow Label: 0x0e4a3
    Payload Length: 44
    Next Header: TCP (6)
    Hop Limit: 1
    Source Address: fe80::e00:ecff:fe11:c601
    Destination Address: fe80::e00:ffff:fee3:3201
    [Source SLAAC MAC: 0c:00:ec:11:c6:01 (0c:00:ec:11:c6:01)]
    [Destination SLAAC MAC: 0c:00:ff:e3:32:01 (0c:00:ff:e3:32:01)]
v Transmission Control Protocol, Src Port: 62412, Dst Port: 179, Seq: 0, Len: 0
    Source Port: 62412
    Destination Port: 179
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1058496507
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1011 .... = Header Length: 44 bytes (11)
  > Flags: 0x002 (SYN)
```

```
0000  0c 00 ff e3 32 01
0010  e4 a3 00 2c 06 01
0020  ec ff fe 11 c6 01
0030  ff ff fe e3 32 01
0040  00 00 b0 02 40 00
0050  03 00 01 01 08 0a
0060  00 00
```

Figure 3-12 *Packet capture of TCP three-way handshake using IPv6 link-local addresses*

In the BGP OPEN message exchanged between spine1 and leaf1, the extended next-hop capability is advertised, confirming that both devices support IPv4 NLRI encoded with an IPv6 next-hop address, as shown in Figure 3-13.

Once all leafs and spines are configured in the same way, an eBGP peering is established between the fabric nodes, as shown in Example 3-19 from the perspective of spine1 and spine2.



```
No.  Time      Source                      Destination                 Protocol  Length  Info
  3 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  BGP      109 NOTIFICATION Message
 11 2023-1…  fe80::e00:ffff:fee3:3201  fe80::e00:ecff:fe11:c601  BGP      167 OPEN Message
 12 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  BGP      167 OPEN Message
 14 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  BGP      105 KEEPALIVE Message
```

```
> Frame 11: 167 bytes on wire (1336 bits), 167 bytes captured (1336 bits)
> Ethernet II, Src: 0c:00:ff:e3:32:01 (0c:00:ff:e3:32:01), Dst: 0c:00:ec:11:c6:01 (0c:00:ec:11:c6:01)
> Internet Protocol Version 6, Src: fe80::e00:ffff:fee3:3201, Dst: fe80::e00:ecff:fe11:c601
> Transmission Control Protocol, Src Port: 61238, Dst Port: 179, Seq: 1, Ack: 1, Len: 81
v Border Gateway Protocol - OPEN Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 81
    Type: OPEN Message (1)
    Version: 4
    My AS: 65500
    Hold Time: 90
    BGP Identifier: 192.0.2.101
    Optional Parameters Length: 52
  v Optional Parameters
    > Optional Parameter: Capability
    > Optional Parameter: Capability
    > Optional Parameter: Capability
    > Optional Parameter: Capability
    > Optional Parameter: Capability
    > Optional Parameter: Capability
    > Optional Parameter: Capability
    v Optional Parameter: Capability
        Parameter Type: Capability (2)
        Parameter Length: 8
      v Capability: Extended Next Hop Encoding
          Type: Extended Next Hop Encoding (5)
          Length: 6
          AFI: IPv4 (1)
          SAFI: Unicast (1)
          Next hop AFI: IPv6 (2)
```

```
0000  0c 00 ec 11 c6
0010  25 cd 00 71 06
0020  ff ff fe e3 32
0030  ec ff fe 11 c6
0040  a5 31 80 18 42
0050  c8 7a 8a 9f da
0060  ff ff ff ff ff
0070  02 65 34 02 06
0080  02 00 01 02 02
0090  78 02 06 41 04
00a0  06 00 01 00 01
```

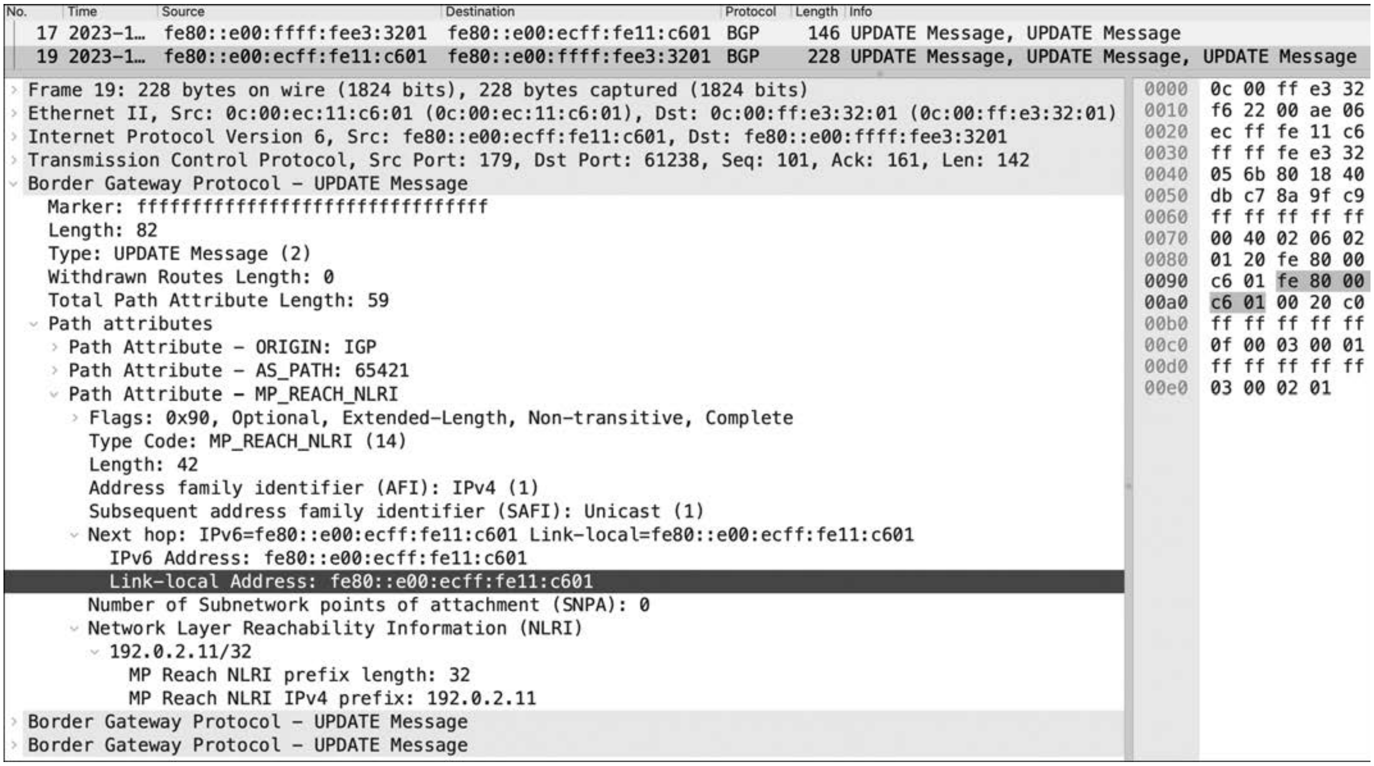
Figure 3-13 *Packet capture of BGP OPEN message from spine1 advertised with extended next-hop capability*

Example 3-19 *Summary of BGP peers on spine1 and spine2*

```
admin@spine1> show bgp summary

Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 1 Peers: 3 Down peers: 0
Auto-discovered peers: 3
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                       0          0          0          0          0          0
inet6.0
                       0          0          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Received/Accepted/Damped...
fe80::e00:36ff:fe96:af01%ge-0/0/1.0       65422        207        205       0       0     1:31:38 Establ
  inet.0: 0/0/0/0
  inet6.0: 0/0/0/0
fe80::e00:bdff:fed8:c901%ge-0/0/2.0       65423        206        204       0       0     1:31:00 Establ
  inet.0: 0/0/0/0
  inet6.0: 0/0/0/0
fe80::e00:ecff:fe11:c601%ge-0/0/0.0       65421        275        273       0       0     2:02:23 Establ
  inet.0: 0/0/0/0
  inet6.0: 0/0/0/0
```

```
admin@spine2> show bgp summary

Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 1 Peers: 3 Down peers: 0
Auto-discovered peers: 3
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                       0          0          0          0          0          0
inet6.0
                       0          0          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Received/Accepted/Damped...
fe80::e00:11ff:fe86:9602%ge-0/0/1.0       65422        207        206       0       0     1:31:54 Establ
  inet.0: 0/0/0/0
  inet6.0: 0/0/0/0
fe80::e00:7dff:fe45:5902%ge-0/0/0.0       65421        211        209       0       0     1:33:18 Establ
  inet.0: 0/0/0/0
  inet6.0: 0/0/0/0
fe80::e00:95ff:feec:8502%ge-0/0/2.0       65423        206        205       0       0     1:31:16 Establ
  inet.0: 0/0/0/0
  inet6.0: 0/0/0/0
```

The last piece of the puzzle is how IPv4 routes are advertised over this IPv6 BGP peering. Since the BGP group is configured to use an extended next-hop for the IPv4 address family, IPv4 routes can be advertised with an IPv6 next-hop address, as shown in Figure 3-14. In this packet capture, leaf1's loopback address, 192.0.2.11/32, is advertised with an IPv6 next-hop address that matches leaf1's respective link-local IPv6 address.



```
No.  Time       Source                      Destination                 Protocol Length Info
  17 2023-1…  fe80::e00:ffff:fee3:3201  fe80::e00:ecff:fe11:c601  BGP      146 UPDATE Message, UPDATE Message
  19 2023-1…  fe80::e00:ecff:fe11:c601  fe80::e00:ffff:fee3:3201  BGP      228 UPDATE Message, UPDATE Message, UPDATE Message
```

```
> Frame 19: 228 bytes on wire (1824 bits), 228 bytes captured (1824 bits)
> Ethernet II, Src: 0c:00:ec:11:c6:01 (0c:00:ec:11:c6:01), Dst: 0c:00:ff:e3:32:01 (0c:00:ff:e3:32:01)
> Internet Protocol Version 6, Src: fe80::e00:ecff:fe11:c601, Dst: fe80::e00:ffff:fee3:3201
> Transmission Control Protocol, Src Port: 179, Dst Port: 61238, Seq: 101, Ack: 161, Len: 142
v Border Gateway Protocol - UPDATE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 82
    Type: UPDATE Message (2)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 59
  v Path attributes
    > Path Attribute - ORIGIN: IGP
    > Path Attribute - AS_PATH: 65421
    v Path Attribute - MP_REACH_NLRI
      > Flags: 0x90, Optional, Extended-Length, Non-transitive, Complete
        Type Code: MP_REACH_NLRI (14)
        Length: 42
        Address family identifier (AFI): IPv4 (1)
        Subsequent address family identifier (SAFI): Unicast (1)
      v Next hop: IPv6=fe80::e00:ecff:fe11:c601 Link-local=fe80::e00:ecff:fe11:c601
          IPv6 Address: fe80::e00:ecff:fe11:c601
          Link-local Address: fe80::e00:ecff:fe11:c601
        Number of Subnetwork points of attachment (SNPA): 0
      v Network Layer Reachability Information (NLRI)
        v 192.0.2.11/32
            MP Reach NLRI prefix length: 32
            MP Reach NLRI IPv4 prefix: 192.0.2.11
> Border Gateway Protocol - UPDATE Message
> Border Gateway Protocol - UPDATE Message
```

```
0000  0c 00 ff e3 32
0010  f6 22 00 ae 06
0020  ec ff fe 11 c6
0030  ff ff fe e3 32
0040  05 6b 80 18 40
0050  db c7 8a 9f c9
0060  ff ff ff ff ff
0070  00 40 02 06 02
0080  01 20 fe 80 00
0090  c6 01 fe 80 00
00a0  c6 01 00 20 c0
00b0  ff ff ff ff ff
00c0  0f 00 03 00 01
00d0  ff ff ff ff ff
00e0  03 00 02 01
```

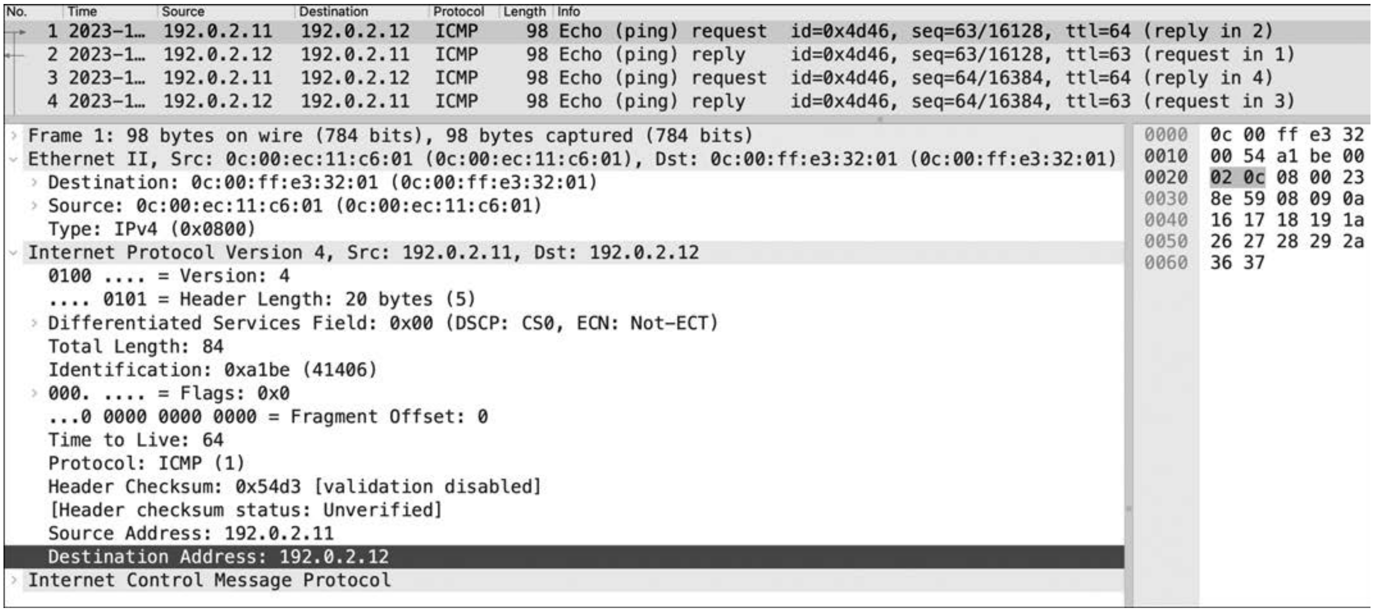
Figure 3-14 *Packet capture of leaf1's IPv4 loopback address advertised with an IPv6 next-hop*

Taking leaf1 as an example again, all remote leaf loopback addresses are now learned with IPv6 next-hop addresses, as shown in Example 3-20, which also confirms loopback to loopback reachability between the leafs.

Example 3-20 *IPv4 loopback addresses learned with an IPv6 next-hop*

```
admin@leaf1> show route table inet.0

inet.0: 3 destinations, 5 routes (3 active, 0 holddown, 0 hidden)
Limit/Threshold: 1048576/1048576 destinations
+ = Active Route, - = Last Active, * = Both

192.0.2.11/32      *[Direct/0] 1d 11:41:30
                    >  via lo0.0
192.0.2.12/32      *[BGP/170] 00:00:27, localpref 100
                      AS path: 65500 65422 I, validation-state: unverified
                    >  to fe80::e00:ffff:fee3:3201 via ge-0/0/0.0
                    [BGP/170] 00:00:27, localpref 100
                      AS path: 65500 65422 I, validation-state: unverified
                    >  to fe80::e00:b3ff:fe09:1001 via ge-0/0/1.0
192.0.2.13/32      *[BGP/170] 00:00:07, localpref 100
                      AS path: 65500 65423 I, validation-state: unverified
                    >  to fe80::e00:b3ff:fe09:1001 via ge-0/0/1.0
                    [BGP/170] 00:00:07, localpref 100
                      AS path: 65500 65423 I, validation-state: unverified
                    >  to fe80::e00:ffff:fee3:3201 via ge-0/0/0.0

admin@leaf1> ping 192.0.2.12 source 192.0.2.11
PING 192.0.2.12 (192.0.2.12): 56 data bytes
64 bytes from 192.0.2.12: icmp_seq=0 ttl=63 time=3.290 ms
64 bytes from 192.0.2.12: icmp_seq=1 ttl=63 time=2.319 ms
64 bytes from 192.0.2.12: icmp_seq=2 ttl=63 time=2.914 ms
64 bytes from 192.0.2.12: icmp_seq=3 ttl=63 time=2.259 ms
^C
--- 192.0.2.12 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.259/2.696/3.290/0.428 ms

admin@leaf1> ping 192.0.2.13 source 192.0.2.11
PING 192.0.2.13 (192.0.2.13): 56 data bytes
64 bytes from 192.0.2.13: icmp_seq=0 ttl=63 time=2.849 ms
64 bytes from 192.0.2.13: icmp_seq=1 ttl=63 time=2.453 ms
64 bytes from 192.0.2.13: icmp_seq=2 ttl=63 time=2.734 ms
64 bytes from 192.0.2.13: icmp_seq=3 ttl=63 time=2.936 ms
^C
--- 192.0.2.13 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.453/2.743/2.936/0.182 ms
```

From the perspective of the data plane, there is no change—the underlay is purely hop-by-hop routing, with a resolution of the Layer 2 address (MAC address) required for every hop. This is already resolved using the IPv6 Router Advertisement messages exchanged between the leafs and the spines, as shown in Example 3-17. Thus, the packet is still an IPv4 packet as shown in Figure 3-15, which is a packet capture of leaf1's reachability to leaf2's loopback address using the **ping** tool, while sourcing its own loopback address.



```
No.  Time      Source       Destination  Protocol Length Info
  1 2023-1…  192.0.2.11   192.0.2.12   ICMP      98 Echo (ping) request  id=0x4d46, seq=63/16128, ttl=64 (reply in 2)
  2 2023-1…  192.0.2.12   192.0.2.11   ICMP      98 Echo (ping) reply    id=0x4d46, seq=63/16128, ttl=63 (request in 1)
  3 2023-1…  192.0.2.11   192.0.2.12   ICMP      98 Echo (ping) request  id=0x4d46, seq=64/16384, ttl=64 (reply in 4)
  4 2023-1…  192.0.2.12   192.0.2.11   ICMP      98 Echo (ping) reply    id=0x4d46, seq=64/16384, ttl=63 (request in 3)
```

```
> Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
v Ethernet II, Src: 0c:00:ec:11:c6:01 (0c:00:ec:11:c6:01), Dst: 0c:00:ff:e3:32:01 (0c:00:ff:e3:32:01)
  > Destination: 0c:00:ff:e3:32:01 (0c:00:ff:e3:32:01)
  > Source: 0c:00:ec:11:c6:01 (0c:00:ec:11:c6:01)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 192.0.2.11, Dst: 192.0.2.12
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xa1be (41406)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x54d3 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.0.2.11
    Destination Address: 192.0.2.12
> Internet Control Message Protocol
```

```
0000  0c 00 ff e3 32
0010  00 54 a1 be 00
0020  02 0c 08 00 23
0030  8e 59 08 09 0a
0040  16 17 18 19 1a
0050  26 27 28 29 2a
0060  36 37
```

Figure 3-15 *Packet capture of leaf1's reachability test to leaf2's loopback, using the* ***ping*** *tool*

Summary

This chapter introduced how BGP can be adapted for a data center, with the benefits it brings, especially for larger-scale data centers. Problems such as BGP path hunting can easily be avoided by using ASN schemes for 3-stage and 5-stage Clos fabrics, or as an alternative, by using routing policies to ensure that sub-optimal paths, which can lead to path hunting, do not exist in the network.

Using eBGP as the underlay and the overlay provides a consolidated and simpler operational and maintenance experience, while continuing to provide vertical separation between the underlay and the overlay by leveraging Junos BGP groups. However, IP addressing for the underlay is operationally challenging and can get complex, very quickly, as the network grows.

With the BGP auto-discovery feature, which uses IPv6 Neighbor Discovery behind the scenes, underlay IP addressing complexity can be eliminated. This also provides an underlay framework that enables easier plug-and-play of fabric nodes, and the capability to automate the underlay without tracking any IP addressing schemes, since all fabric-facing interfaces are configured the same way.

Index