Brought to you by:

apstra

Intent-Based Networking

for dummies

A Wiley Brand

Understand what IBN is and is not

Develop an IBN architectural framework

Transform your network operations with IBN

Apstra Special Edition

Jeff Doyle

About Apstra

Apstra is a multinational software company delivering a unified solution to automate the architecture and operations of the data center network. Apstra's flagship product, AOS, empowers customers to safely and quickly remove complexity by abstracting knowledge from the network and modeling behavior through the build, deploy, operation, and validation phases. AOS uses Apstra's advanced intent-based analytics to continually validate the network, thereby eliminating complexity, vulnerabilities, and outages resulting in a secure and resilient network. Apstra, founded in 2014 by established and proven networking industry visionaries David Cheriton, Mansour Karam, and Sasha Ratkovic, is headquartered in Menlo Park, California, with other offices worldwide.

Intent-Based Networking

Apstra Special Edition

by Jeff Doyle

for dummies®
A Wiley Brand

Intent-Based Networking For Dummies®, Apstra Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN: 978-1-119-68377-3 (pbk); ISBN: 978-1-119-68366-7 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Editor: Carrie Burchfield-Leighton
Development Editor: Ryan Williams
Editorial Manager: Rev Mengle
Acquisitions Editor: Ashley Coffee
Business Development Representative: Karen Hattan
Production Editor: Mohammed Zafar Ali

Table of Contents

Introduction

Digital transformation is all about applications and agility. Modern digital services are usually built from multiple applications — especially in this age of microservices — and both your staff and your network must be optimized for quickly creating and deploying new services, for changing services at the drop of a hat, and for quickly scaling applications that can experience 50 to 100 percent yearly growth.

According to a 2018 McKinsey survey, 68 percent of respondents' objectives were digitizing the organization's entire operating model; less than half had a more limited objective of either launching new products or services or interacting with external partners through digital channels. The same survey reported that less than 30 percent succeed, with organizations of fewer than 100 employees reporting a successful digital transformation, 2.7 times more often than organizations with more than 50,000 employees.

Research shows that digital transformation is a huge endeavor. The last thing you need is for your network to get between your innovative new services and your customers, employees, and partners.

Intent-based networking (IBN) has become a hot buzzword in the networking industry, with marketing departments at all sorts of vendors waving the "intent flag." Some have legitimate products, some have cobbled together bits and pieces out of their product portfolios and called it an IBN solution, and some supposed IBN products perform only a part of what a real IBN system (IBNS) does.

About This Book

This book waves away the fog to provide you with a clear understanding of what IBN really is. You look at what *intent* means in the context of network operations and how an IBNS applies that intent to a network across its entire life cycle. You also delve into what features and characteristics an IBNS requires to fulfill its mission. You look at practical examples and testing of IBN before circling back to the benefits, just so you leave with a good feeling about the whole thing.

Icons Used In This Book

You may notice some little pictures in the margins of this book. Some you can ignore; some you may want to peruse more closely. This section helps you make your determination and maybe save a little time.

This icon gives you a little extra help, saves you time, or may even save you money. Check them out if you're not in a hurry.

This icon emphasizes an important fact. Keep this in mind.

Pay more attention to these icons. You want things to keep running well and with a minimum of disaster, right? These icons are suggestions of things to avoid those disasters.

Isn't this book all technical stuff? Well, not exactly, but some information is more technical than the rest. If you aren't a super techie, you can skip this info.

Beyond the Book

This book helps you discover more about IBN, but if you want resources beyond what this book offers, I have some insight for you:

- **Download a free eBook on the benefits of IBN in the data center.** `https://go.apstra.com/ebook-self-operating-data-center`
- **Download Apstra's whitepaper on the IBN architecture.** `https://go.apstra.com/white-paper-architecture-overview`
- **Dive deeper into the requirements for a good IBN system.** `https://tools.ietf.org/html/draft-clemm-nmrg-dist-intent-03`

IN THIS CHAPTER

- Perusing the challenges of digital transformation
- Reviewing the high-level functions of IBN
- Evaluating the different levels of maturity of an IBNS
- Differentiating a true IBNS

Chapter 1
Expressing Intent and Seeing the Basics of IBN

Intent-based networking (IBN) is far more than just network management. The fault, configuration, accounting, performance, and security (FCAPS) management framework is all part of an IBN system (IBNS). But those aspects are just capabilities, not IBNS itself.

Okay, then what is it? This chapter gives you a bit of the challenges of transformation and then tells you why IBN is beneficial to this transformation.

Looking at the Challenges of Digital Transformation

Multiple industry studies indicate that by the end of 2021, organizations will be *three times* more likely to fail in their digital business transformation if they don't adjust their operational practices. But most IT organizations — around 82 percent — struggle to just keep running, leaving meager funding for innovative development.

Similarly, network architects spend more than 50 percent of their time serving as top-tier operational support when they should be focused on staying on top of technology trends and developing three- and five-year plans incorporating those trends.

Data center technologies have evolved to support digital transformation. Micro-segmentation, containerization, microservices, and service virtualization all contribute to building agile digital environments. Orchestration systems ease the operational burden, at least for storage and compute, by operating on an abstracted model of the physical systems.

And although network virtualization technologies such as *Virtually Accessible LAN* (VXLAN) and *Ethernet VPN* EVPN support highly mobile digital end systems and applications, network operations lag so far behind on the transformation curve that it often inhibits change instead of promoting it.

More often than not, a network's operational problems spring directly from humans interfaced too close to the systems.

The human interface

Humans are slow, expensive, error prone, and inconsistent. They're irreplaceable when interacting with systems at a level where insights are unique, but when interfaced directly to network systems via CLIs, on-the-fly scripting, or web-based configuration management tools, the systems are vulnerable to small mistakes that can have enormous costs to the business. Strong change management polices reduce error rates, but at the price of even slower change processes.

THE OPEX CHALLENGE

Once upon a time, organizations dealt with soaring operational expenses (OPEX) by reducing staff — leaving the remaining personnel to pick up the slack. But reducing operations staff when IT is vital to your business just compounds your problems. Today OPEX reduction is about reducing the time required to perform the countless individual tasks of operating a network.

The thing is, humans are marvelously talented at pattern recognition and have mad skills at developing unexpected solutions from available data (you know, what you usually call thinking outside the box, or innovation).

What humans are *not* good at are mundane, repetitive tasks over a long period. People get bored and make mistakes. One mistake too many and your company is on the cover of the Wall Street Journal for all the wrong reasons.

The human interpreter

There's a linear progression from business intent to a successful network process. In the middle is an essential human translation layer: the network architect. This person consumes vast amounts of coffee and meeting time, takes business intent as an input, translates that to technical intent, and outputs workable network configurations.

Just like the human interface in the operations center, the human interpreter is irreplaceable for the abstract parts of the job but is slow and error-prone at the lower task-oriented part of the job. And they're not called network architects for nothing. The job has a distinct design element to it that, while extremely important, can drift into individual styles that introduce inconsistencies to your network.

Individualism matters if you're Michelangelo or Miles Davis. Individualism in network design can be dangerous.

Inadequate automation

Whether it's handling repetitive operational tasks or generating and pushing new configurations, operators and architects have long recognized the value of automation both as a labor-saving tool and as a means of reducing human error.

Most automation tools, from old Tcl and Python scripts to modern provisioning software like Ansible, take input for a specific task and output configurations specific to your current network. If your network changes — you add new features, or you change vendors — your scripts must change with your network. And running a multi-vendor network can mean maintaining multiple scripts that do the same thing.

Even if you build a large library of scripts over the years, they're likely to be disjointed and more oriented to Day 0 provisioning than to managing the full life cycle of your network.

Data overload

It's not that most of your swarm of data is unimportant. It's that different data is important at different times. How do you sort through it all, and how do you determine what data points are directly related? What do the interrelationships tell you about the health of your network? When an anomaly appears, how do you identify the data that points to a root cause?

The challenge in a transformative network isn't the flood of data. It's being able to run analysis on the data, within context and in near real time.

Stale documentation

Maintaining up-to-date documentation is a challenge even for most legacy networks. Networks supporting modern digital services change not just day-to-day or hour-to-hour, but often second-to-second. Humans, and even many automated documentation systems, simply can't keep up. Yet without an accurate map of the physical and logical infrastructure, you can't safely maintain the rates of change demanded by digital services.

Does that seem like circular reasoning? It is, unless your network is self-documenting.

Making the Transformation with IBN

IBN smooths many of the speed bumps standing in the way of successfully transforming and operating your network by supporting rapid, at-scale changes, making your network more autonomic throughout its life cycle, and providing insights into your network that are always up-to-date. You can manage what requires automation, make your system standardized and reliable, and ensure you're free to move and adjust heading into the future.

While *capital* expenses (CAPEX) are certainly nothing to sneeze at, the budget IT executives continually struggle to get under control is *operational* expenses (OPEX). IBN is all about getting OPEX under control in a variety of ways.

In transforming the way you operate your network, with IBN you'll see the following benefits:

- **Managed complexity:** Break operational tasks down to their simplest elements and automate them, based on expected outcomes.
- **Managed risk:** Eliminate human error in the flow from expressed intent to creation and deployment of specific configurations.
- **Managed "data fog":** Get actionable insights into the massive big data telemetry your network is throwing at you, eliminating the heavy operational expense of extracting only the data you need at a specific moment.
- **Increased reliability:** Operations that run 24/7/365 make maintenance windows increasingly difficult to schedule. Network changes under IBN are markedly faster and can often be performed in production.
- **Standardized network segments:** Use validated, best-practice blueprints to quickly stamp out reliable, industry-standard network segments.
- **Accelerated agility:** Everyone talks about network agility. All it means is the ability to adapt to changes and new applications without major structural changes. Agility is directly tied to operational cost savings.
- **Freeing your experts:** IBN means your architecture team spends less time fighting fires and more time working on strategic initiatives.
- **Surveying your options:** Dealing with the quirks of individual network operating systems unnecessarily lengthens deployment times. They can even force architects to adapt to vendor capabilities instead of what's best for the business. IBN puts design first and deals with vendor specifics in the background.
- **Moving from days to minutes:** Imagine that you need to deploy ten new leaf switches in their data center. After racking and cabling everything, you also need to upgrade the operating systems. Next, develop, validate, and deploy new. Finally, it's time to run your acceptance tests. Altogether, bringing the new switches online can take multiple days. After moving to an IBN system (IBNS), the same project — from design to deployment and acceptance testing — takes 20 minutes.

FLYING ON AUTO-PILOT

Simply put, IBN lets software do what software does best — perform mundane operational tasks quickly, accurately, and cheaply.

Boeing 777 pilots report that they spend approximately seven minutes manually flying the plane on an average trip. They input the flight parameters (intent), and the plane's internal systems translate that expressed intent into all the operational minutiae necessary to complete the flight. Along the way, the systems constantly adjust to ensure constant compliance with the expressed mission objectives.

The 777 pilot isn't eliminated, and the pilot's role isn't trivial. To the contrary, the pilot is relieved of the simple minute-to-minute tasks and can better apply his expertise to the overall mission. The pilot can always take over in the event of a system failure. Increasingly, though, the system can also take over in the event of a pilot error.

The parallels to IBN are powerful. You express your technical intent, and the system takes over the mundane tasks of not only deploying and operating your network in compliance with your intent but also constantly ensuring that the network doesn't deviate from intent. Your network is its own auto-pilot.

Digging Deeper into IBN

IBN takes your network — regardless of the specific vendor or operating system of your network devices — from piecemeal node-by-node management to an autonomic network. The system self-operates, self-adjusts, and self-corrects within the parameters of your expressed technical objectives.

Those expressed technical objectives are your *intent*. Intent is a declarative statement of expected outcomes. Traditionally, network architects conduct engineering meetings where intent is the input and executable, device-specific configurations are the output. Outputs can also include validation of the network configurations and operational parameters for continued compliance.

That's where an IBNS comes in. An IBNS takes your intent as input, translates your intent first into actionable policies for your network, and then executes device-specific configurations.

It validates the outcomes and actively monitors the network for intent compliance, adjusting network parameters as needed to ensure compliance throughout the network life cycle. The result, as shown in Figure 1-1, is vastly simplified, highly reliable network operations.

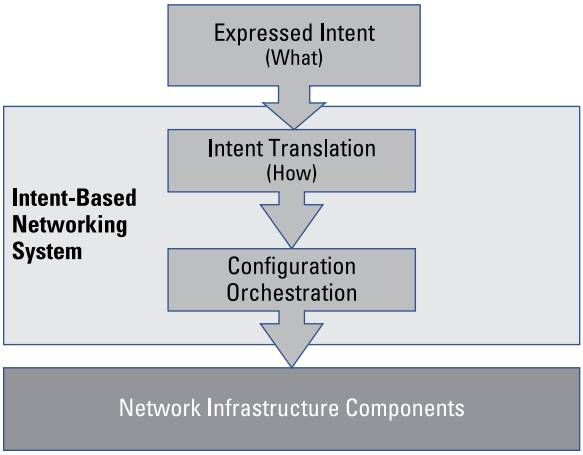

FIGURE 1-1: A 30,000-foot view of an IBNS.

You say what, it says how

Intent in the context of IBN is a declarative statement. How very academic. An easier way to define it is that you tell an IBNS *what* you want, and the system decides *how* to do it. Gone is the human interface translating expressed intent into executable configurations, injecting individual interpretations (read: inconsistencies) and possibly mistakes into the configuration. The IBNS performs the translation, consistently and accurately.

Take this list, for example:

- Deploy an L2 network interconnecting VMs for application X in the data center.
- Ensure 3:1 oversubscription or better on all links.
- Isolate the network from all other tenants in the data center.
- Provide external access through routers Y1 and Y2, applying security policy Z.

The IBNS accepts this declaration of operational requirements and references a blueprint consisting of services, resources, and a reference design. The network validates the requirements and creates configurations specific to individual nodes. Finally, the network automatically pushes this information to the physical network infrastructure.

Information flows

Figure 1-1, earlier in this chapter, shows the flow of information from your expressed intent (desired outcomes) through the elements of the IBNS to the physical network. This isn't enough. IBN has to know these factors as well:

- The topology of the physical infrastructure
- The vendor and operating system version of each network node
- The resources available at each node and link

So, information not only flows down from the IBNS to the infrastructure but also flows up from the infrastructure to the IBNS. Figure 1-2 shows you the system that must collect the information necessary to represent the network and monitor network resources and states.

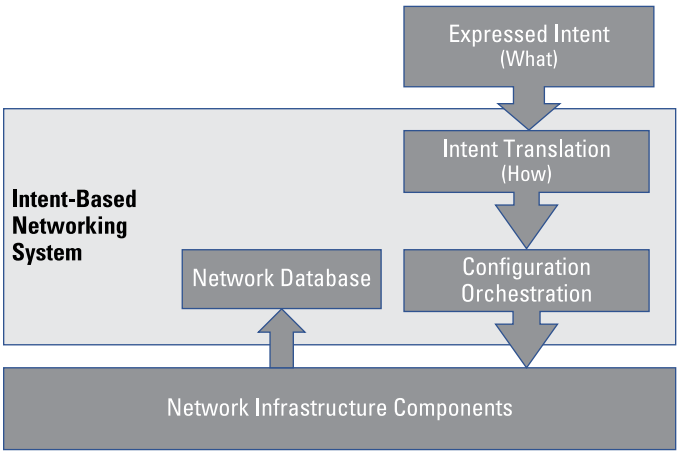

FIGURE 1-2: Information must be collected from the network to appropriate data stores.

This two-way flow extends IBN beyond mere deployment to network support through its entire life cycle.

Following the full life cycle support

Four phases define the network life cycle:

1. **Design.**
2. **Build.**
3. **Deploy.**
4. **Validate.**

You can roughly judge the maturity of an IBN product by how thoroughly it supports the network in all four of these phases. But you can evaluate an IBN offering even more effectively by using an objective, fact-based taxonomy to categorize its level of development. Figure 1-3 shows a taxonomy for four levels of IBN deployment. Each level is increasingly impactful on business operations.

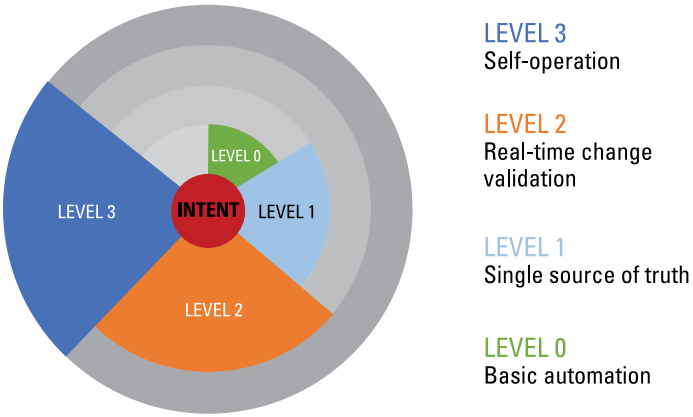

FIGURE 1-3: The IBN taxonomy.

Level 0: Basic automation

At Level 0, a basic IBN solution can generate configurations based on declarative statements and push them out to network nodes. Although there's probably some "upward" information flow from the network devices, IBN at this stage is likely to be limited to a single vendor. The information can be message-centric, rather than data-centric. What's missing is a single source of truth.

Level 1: Single source of truth

A single source of truth, illustrated in Figure 1-4, makes IBN data-centric rather than message-centric. The single source of truth stores both expressed intent and of recursively updated network state.

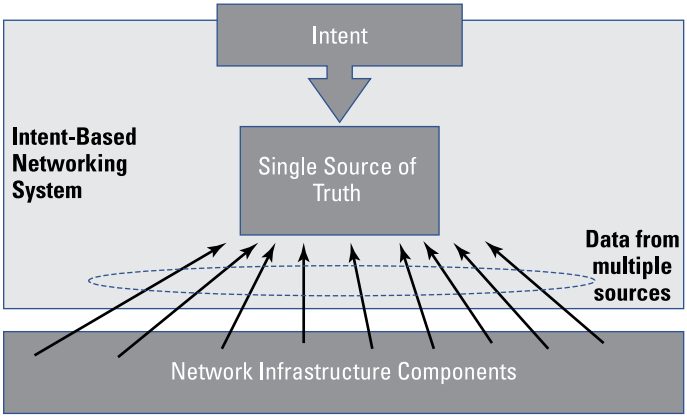

FIGURE 1-4: Level 1 IBN requires a single source of truth.

A single source of truth can query the database (instead of the network) at any time for either existing state, such as "What is the status of interface X on node Y?" You can also ask potential states, such as "What is the impact on bandwidth utilization if I take node Z offline?"

Level 2: Real-time change validation

Real-time change validation requires an addition to the information flows from Figure 1-2 in the earlier section "Information flows." Real-time, closed-loop telemetry continuously verifies up-to-the-second network state against expressed content. Take a look at Figure 1-5 for an example. Real-time change validation requires continuous verification of network state against expressed intent.

Networks continually undergo both planned and unplanned change, and each change must be evaluated against your expressed intent. Think of variation from intent as *intent drift*. At the very least the network must alert you when it has gone out of compliance. At best, the network should self-adjust without operational intervention. At this point, you enable the highest level of IBN, which is self-operation.

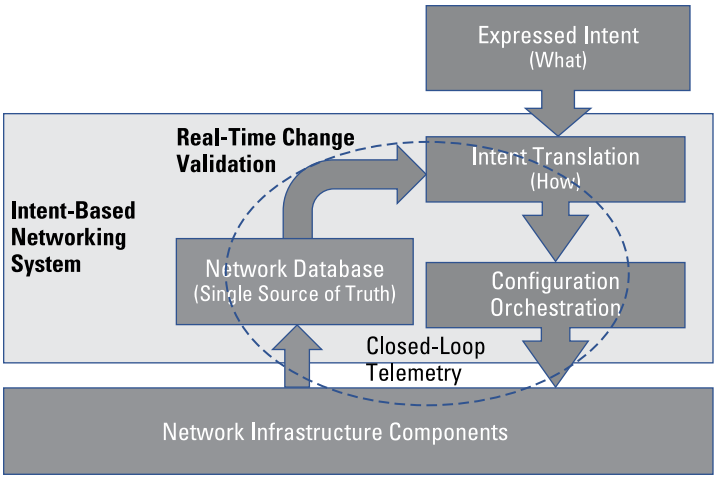

FIGURE 1-5: Real-time change validation requires closed-loop telemetry.

Level 3: Self-operation

Only when an IBN solution solidly supports *all* of Levels 0 through 2 does it move to the final level of achieving network autonomy. Level 3 allows the network to learn and adapt to network fluctuations. While the previous levels provide *intent fulfillment*, Level 3 reaches *intent assurance*.

REMEMBER

Intent assurance is the ability to reason about a system's state by employing closed-loop validation in the presence of an inevitable change. At Level 3, your self-operating network frees your operational staff members from low-level, mistake-prone tasks and empowers them to monitor your network holistically. Operations that used to take hours or days are now performed in seconds, giving you an adaptive, agile network.

Understanding What IBN Is Not

IBN is, unquestionably, a popular industry buzzword. Your challenge is to see beyond the fluff and evaluate IBN solutions based not only on what they are — as I discuss in previous parts of this chapter — but also by what it is *not*.

Interestingly, all the functions discussed in this section can be and probably are a part of an IBN solution. What you have to be wary of is a solution that performs one or a few of these functions and claims that that, alone, is IBN.

IBN is not automation

Automation, from home-grown scripts to platforms like Ansible and Puppet, are essential to the fast, reliable operation of a network. It's also an essential element of IBN. But automation software says nothing about expressed intent and doesn't by itself maintain a data store of network information to act on. You can automate bad decisions just as well as good ones.

IBN is not configuration management

A Level 0 IBNS may look like just a fancy configuration management platform that translates intent into practical configurations. Such a system falls far short of significantly improving your operational effectiveness.

IBN is not SDN

You may be thinking that IBN is just a form of software-defined networking (SDN). But SDN, in its usually understood role, performs only a part of what an IBNS does.

SDN maintains an abstracted model of the physical network. It takes generic configuration commands as input and pushes device-specific configuration as its output. But that's all. SDN contains no translational element to convert intent into generic configuration and it has no capability for continued compliance verification and adjustment.

TIP

Like automation and configuration management, SDN is an element of IBN, but is not itself IBN.

IBN is not orchestration

Orchestration helps all of your IT systems — compute, storage, and network — act in sync to accomplish your higher IT objectives. IBN, as again the name implies, is concerned just with your network. That said, a good IBNS should integrate with your orchestration system so that orchestration can become a source of declared intent.

IBN is not a policy engine

Policy engines can both "push" policies to network nodes and "pull" information from the nodes to continually verify correct policy enforcement. But the translation of intent into an actionable "how" is missing. Policies just govern aspects of the network, such as forwarding, security, and prioritization. A policy engine can use control loops to enforce these policies, but it has no concept of desired outcomes. You have to work those out yourself and specify in detail the rules to implement and enforce the policies.

IN THIS CHAPTER

- Learning about intent fulfillment and intent assurance
- Discovering why idempotency is important to an IBNS
- Identifying your single source of truth (SSoT)
- Gazing through your simple pane of glass
- Speaking the truth

Chapter 2
Looking at the Characteristics of IBN

Everyone and her dog claim to have an intent-based networking (IBN) solution these days. Some are valid (the solution, not the dog — all dogs are good and valid). Some solutions are workable but are limited to a single vendor. Other solutions fulfill your expressed intent but don't continue to validate intent after deployment. And the rest of the solutions are just cobbled together from multiple products designed to do other things.

To help you clear away the fog, this chapter presents the characteristics and capabilities you should look for in any IBN solution.

Understanding the Fundamental Aspects of IBN

A true IBN system (IBNS) contains two fundamental qualities that make it more than just a fancy configuration management platform: *intent fulfillment* and *intent assurance*.

Intent fulfillment

Intent fulfillment is the more obvious of the two fundamental IBN characteristics — you say what you want, and the IBNS fulfills your requirements. Accuracy and consistency are essential in translating your intent to a working service, and that requires a well-thought-out architecture under the hood. While the terms and details may vary from one implementation to another, the IBNS architecture should support the elements in this section. I also discuss these in more detail in Chapter 3.

Hit the reference section

An IBN applies a *reference design* (RD) to your expressed intent. The RD contains a set of best practice rules for fulfilling your requirements, specific to what you're asking. For example, if you need to build an Ethernet VPN (EVPN) overlay, an RD specific to that requirement is applied. The RD knows what variables, design options, internal process rules, configuration syntax for all supported vendors, and validators for checking the resulting configuration is needed.

It's so abstract . . .

An IBN includes a database of *abstractions*, such as details of the generic kinds of devices required to fulfill your intent. For example, the abstraction may include the number of switches you need, how they're connected, the number of ports, and the port speeds. This element doesn't list actual products and doesn't list anything outside of what is required to fulfill the intent. In this way, the results are a best fit to your intent, not a best fit to any one vendor solution.

Take inventory

IBN also includes an *inventory* of what is actually available to you that fulfills your abstraction. The most apparent part of this database is an exhaustive list of vendors and models. This list enables you to either select devices you already have or that helps you to know what vendors offer the right models for you. But there's more than just devices. Other resources available to you are also

maintained here, such as Internet protocol (IP) address ranges, autonomous system number pools, and virtual LAN (VLAN) IDs.

Review the blueprints

A *blueprint* pulls everything together from the reference design, the abstraction, the inventory, and the existing network state to push a valid, verified, repeatable service to your network.

Intent assurance

You can't just deploy a network service — not even a thoroughly verified one — and walk away from it. Networks change for all sorts of reasons, and that potential for change is why intent assurance is vital. You need to know if a service diverges from intent.

The validators in the reference design are essential for providing assurance before a service is deployed, while it is being deployed, and for the full length of time the service remains deployed. Think of them as the tattletale that lets you know when somebody does something naughty. In many cases, variations from intent can be corrected automatically. When your intervention is required, the validators should provide you with a clear analysis of the problem so you can quickly bring the service back in compliance with little or no downtime.

Chapter 4 examines intent assurance through intent-based analytics (IBA) and root cause analysis (RCA).

Idempotency

We've all heard the time-worn cliché that the definition of insanity is doing the same thing over and over and expecting different results. Idempotency is sort of the other side of that coin: Doing the same thing over and over and expecting the *same* results.

That's not so crazy.

Networks and services evolve over time, and that means changes that you make today may have a different effect than the same changes made weeks or months later. An IBNS *must* have current insight into the network so you can expect any two or more identical changes you make, at any time in the life cycle, have the same result. Hence, idempotency.

REMEMBER

An idempotent operation is one that you can perform repeatedly, and the results are always the same.

This quality stems from a combination of the full life cycle validators that make intent assurance possible and from up-to-the-minute network telemetry. Without this assurance, the effect of a failed change could be enormous.

Single Source of Truth

Intent assurance and idempotent operations are impossible if you're working from many sources of truth. Each element of your network has its own unique "perspective" of the network. In fact, each element probably comprises a set of states, and each of those states can be considered a specialized view of the network. These views — some set of relevant data, operational states, and telemetry — provide a source of truth.

Not only do these sources of truth vary across elements, but also in a multi-vendor network, functionally similar elements may represent their view of the network differently. And some vendors are even moving away from integrated operating systems, supporting different files within a device for different functions.

A *single source of truth* (SSoT) means that all network operations act on a single data set. Figure 2-1 shows a sneak peak of the IBNS architecture (I discuss this more in Chapter 3). This blueprint takes information from the infrastructure and from other entities within the IBNS. The system consolidates all the inputs into a single dataset (the single source of truth) and views the network entirely from that perspective.

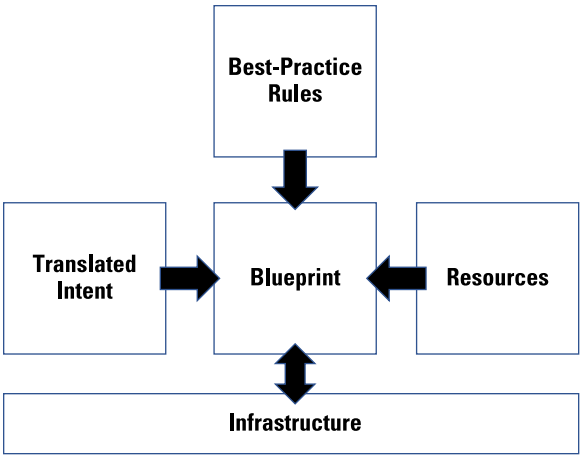

FIGURE 2-1: The IBN works from a blueprint that represents a single source of truth.

Simple Pane of Glass

You may have heard of a *single pane of glass* in reference to management and control software. It's an important concept and a valuable benefit of having a single source of truth: You can see your entire network from a single, consistent perspective.

That view is great when you want a holistic view of your entire network, but what about when you want to see just one specific part of your network? In this case, you not only want a single pane of glass, but also you want a *simple* pane of glass. In other words, you should be able to say what you want to see, and see only that and nothing more. That concept sounds simple, but it's a tall order. This capability extends intent beyond fulfillment and assurance and allows you to express intent when you're viewing your network.

The value of a simple pane of glass becomes particularly apparent when you're troubleshooting. Instead of sorting through a massive cloud of irrelevant data, you can quickly zero in on the relevant information you need to pinpoint the root cause. Sweep all the irrelevant material aside and turn that glass into a microscopic lens.

Speak the Truth

Not all IBNSes are the same. In fact, not all supposed systems that claim to be intent-based really are. A true IBNS must not only fulfill your expressed intent but also must assure that your network, regardless of vendor, remains in compliance with your intent across the full life cycle.

Your IBNS must serve as a single source of truth and provide insights into the massive data that goes into the SSoT so that you can zero in on what you are interested in and nothing more.

IN THIS CHAPTER

- Examining what an IBN workflow looks like
- Getting a high-level view of IBN architecture
- Discovering reference design
- Learning how the components of an IBN system work together

Chapter 3
Detailing the IBN Architecture

After you see what intent-based networking (IBN) is (and is not) and what characteristics an IBN solution should have (I cover these topics in Chapters 1 and 2), the next step is translating all that into a practical IBN architecture. The architecture presented in this chapter shows how IBN characteristics can be implemented and delivered. This method certainly isn't the only way to do it, but what you read here is proven in large scale production networks.

Making the Big Stuff Small

Humans take an objective-oriented (intent!) view of a task, such as "I want a VLAN connecting servers A, B, C, and D." It may seem like a simple operation, but in fact it's an abstract, "big-picture" view of what really needs to be done. The number one job of an IBN architecture is to break these kinds of tasks down into their constituent tasks, then recursively break those constituent tasks down until it gets to the very simplest sets of steps, information, and variables needed to instruct the relevant network

components. It's kind of like telling your kids to clean their rooms, then reminding them of all the small things that go into that.

A step-by-step expression of a workflow to accomplish this task is as follows:

1. **Define the intent.**
2. **Translate intent into a series of prescriptive network changes.**
3. **Verify that the changes are valid before deploying them.**
4. **Deploy the verified changes.**
5. **Monitor network state to ensure continuous compliance with the intent.**

Figure 3-1 shows a first pass at breaking this high-level workflow down into something that can be implemented. The figure also shows an overlay on the workflow with the Design, Build, Deploy, Validate life cycle (I talk more about this in Chapter 2). The difference between the workflow and the life cycle is that the overlay defines phases of a service's life. The workflow itself defines steps for bringing a service to life.

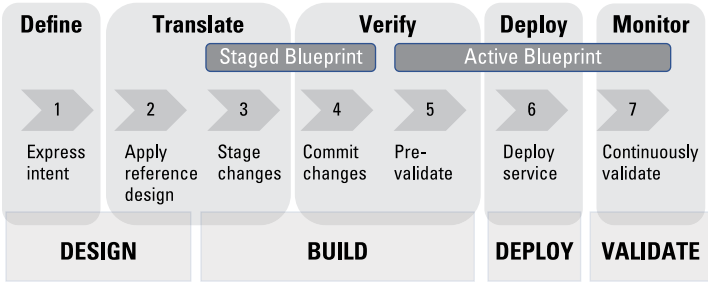

FIGURE 3-1: The IBN workflow touches every phase of the service life cycle.

Notice that a *blueprint* is a part of the workflow. This is a first indication that there must be an information repository with which the workflow steps can work. The blueprint is the beginning of constructing the IBN architecture itself.

REMEMBER

The architecture and functions described in this section fit the Design, Build, and Deploy phases of the service life cycle. The blueprint isn't just for getting a service up and running, though. The IBNS continually uses the validation mechanisms defined in

the blueprint to ensure constant compliance to intent. I discuss these processes in Chapter 4.

Getting a High-Level View of IBN

At the heart of a conceptual IBN architecture, shown in Figure 3-2, is another blueprint, which contains all the information needed for deploying and maintaining a system based on expressed intent.

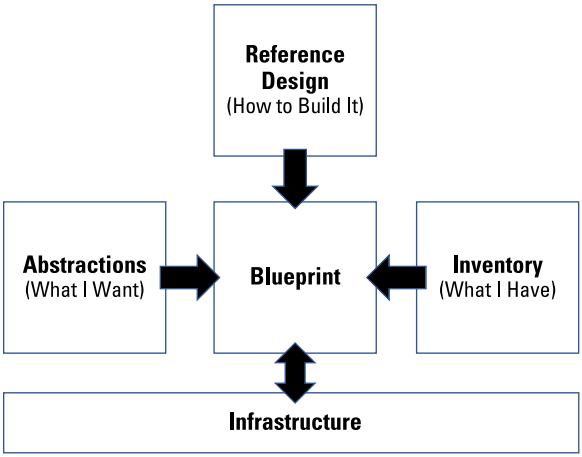

FIGURE 3-2: A conceptual IBN architecture.

A number of information sources contribute to the building of the blueprint:

- **Abstractions** are generalized expressions of the service — that is, what you want.
- **Inventory** is a listing of the resources you have available to you. Notice that in Figure 3-2, infrastructure is connected to the blueprint with a two-way arrow. The blueprint not only pushes configuration to the infrastructure, but also it pulls necessary telemetry from the infrastructure for validation and analysis.
- **Infrastructure** is the set of managed physical and virtual elements in your network.
- **Reference design** is the behavioral contract that governs the service you want to deploy.

The Reference Section

A *behavioral contract* isn't what you expect from the teenager who just asked for her own car. Instead, this element is a best-practice definition of the rules for creating a service. The IBNS holds a library of *reference designs* (RD), and when you express your intent to build a service, the relevant RD is consulted.

For example, if you want to deploy a Virtual Extensible LAN (VXLAN) service, use a VXLAN reference design, as shown in Figure 3-3. That RD contains all of the general rules for configuring VXLAN, all the valid design options, a detailed listing of all tasks for configuring VXLAN, and all the variables that must be input to create a VXLAN configuration.

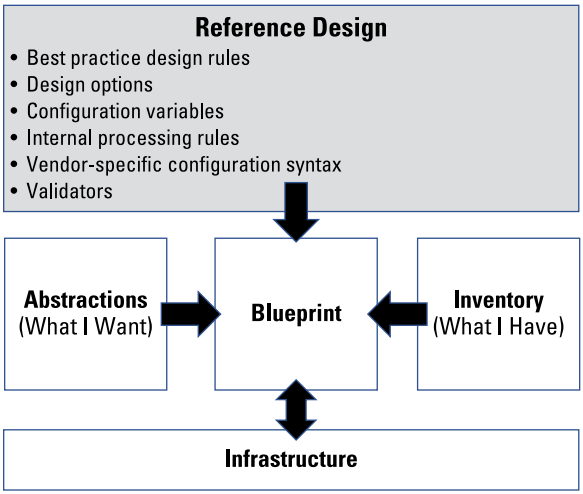

FIGURE 3-3: The reference design contains rules for configuring and deploying a service.

The RD also contains essential information for the IBNS processing itself. If the IBNS uses a graph model in the background for visualizing the service, the RD specifies the nodes and links (along with associated information in each) to add to the graph in order to represent the service.

Beyond the general best-practice rules, the RD contains the specific configuration syntax for every network operating system

the IBNS supports. For example, the VXLAN reference design may contain all the VXLAN configuration syntax for Cisco NXOS, Cumulus Linux, Arista AOS, and Juniper JUNOS. Also associated with the network operating systems are vendor-specific piece of information such as device requirements and configuration limitations. Finally, the reference design defines validators that are used to

- Pre-validate a configuration before deployment (see Step 5 in the earlier Figure 3-1).
- Validate the configuration after it's deployed (see Step 6 in the earlier Figure 3-1).
- Continuously validate the service against intent throughout the service lifetime (see Step 7 in the earlier Figure 3-1).

Validation is detailed in Chapter 4, but you can easily see that the service-specific validators defined in the reference design are fundamental to the operational benefits of IBN. The reference design is also the key element in the IBN architecture. It translates expressed intent into deployable configuration, verifies the configuration through a firm set of best-practice rules, and sets the procedures for continuous compliance to your intent.

Making It Abstract

Every good design begins with an abstraction of the network or service. If you're starting with a single vendor's capabilities and designing from there, you're putting too many constraints on what you may have. The abstract model of the network or service contains everything you need to implement, with neither the constraints of a single vendor nor the distractions of irrelevant features found in all network operating systems.

TIP

Think of this element like your shopping list for the perfect menu, even if you need to go to a million grocery stores for the ingredients.

REMEMBER

The abstraction isn't an element of the IBN system itself; instead it's a category of elements, as depicted in Figure 3-4. The example abstractions are for data centers, but the same concept can apply to cloud, campus, enterprise, service providers, or any other network construct. There's no limit.

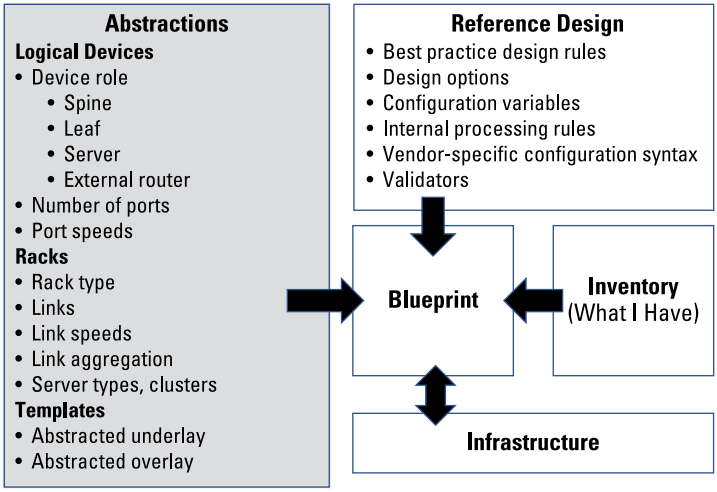

FIGURE 3-4: The abstractions of the IBNS are generic specifications of the details.

Keep it logical

Start building your abstraction by specifying your requirements for each device comprising the network or service you're designing. For each device, this includes its role (such as spine switch, leaf switch, L2 server, or L3 server), the number of ports you require, and the port speeds. Note that the role the logical device plays is only relevant to your design. The same device may play other roles in other services. Similarly, the number of ports doesn't necessarily mean the number of ports in the physical device fulfilling the role, but just the ports needed for your design.

Off the rack

The rack also corresponds to the network or service you're designing. It may correspond exactly with a physical rack of equipment in your network, but it has the same characteristics. One top-of-rack switch or two? What uplinks and uplink speeds? How many servers, what kinds, and what connection speeds? IPv4, IPv6, or both?

Creating templates

The logical device and rack specifications could go directly into the blueprint. But what if you want to use the same specifications

more than once? By using the device and rack specifications as input to a *template*, as shown in Figure 3-5, you have a sort of rubber stamp that you can use over and over, stamping out as many identical designs as you need. Of course, you can use different templates to create different blueprints. You have the flexibility.

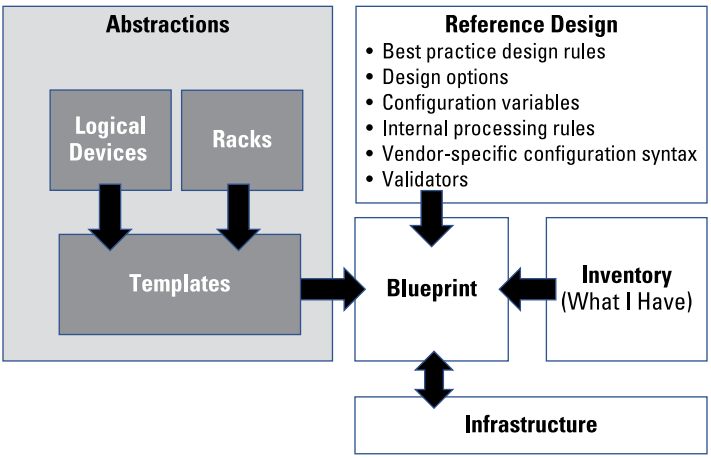

FIGURE 3-5: The templates bring the logical underlay and overlay together.

More importantly, the template serves as the abstracted model for the underlay (3-stage leaf and spine, 5-stage super-spine, campus hierarchy, and so on) and the abstracted model for the overlay (VXLAN or EVPN, for example).

You can also define import and export policies, security zones, external connections, and external protocols under the template.

Taking Inventory

Lined up opposite of your abstractions (what you really, really want) are the realities (what you really, really need to learn to work with) in the IBNS's inventory, shown in Figure 3-6. The *inventory* is a library of profiles that you can pull up and match against what you need. The flexibility here is that you can use it not only to apply your design to your existing physical infrastructure but also to help you determine the right kit to buy.

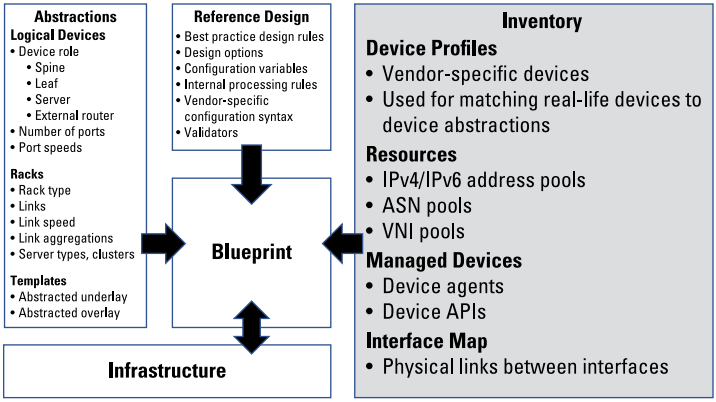

FIGURE 3-6: The inventory is a library of physical devices and resources.

Make a profile

A *device profile* is a detailed specification of a vendor switch, down to the model, operating system version, port count, form factor, silicon type, memory, and capabilities. A good IBNS should have an extensive, regularly updated library of device profiles and the ability for you to add to the library.

REMEMBER

You can dig into the library to find existing switches in your network, or you can match it against your logical device abstractions to see what models are a best match.

Dealing with the elements

Managed elements are the actual devices in your network managed by the IBNS, selected from the device profile library. It lists such things as the device's profile, the management IP address of the device, the serial number, the operating system version, and the hostname.

TIP

Managed elements can also include logical elements such as VSphere switches.

Get out the maps

The *interface map* specifies, for each device in your design, what ports are connected to what other devices' ports, and the speed of each.

Resources

Resources are a pool of addresses and identifiers that the IBNS can use, such as

- IPv4 addresses
- IPv6 addresses
- 16-bit autonomous system numbers (ASNs)
- 32-bit ASNs
- Virtual network identifiers (VNIs)

You can tell the IBNS what pools it can use — such as your publicly assigned IPv4 address range or a range of prate IPv4 addresses — and IBN takes care of assigning addresses and identifiers out of the pools you give it.

Looking at the Blueprint

After you've assembled all your raw materials from the abstractions, reference design, and inventory elements, IBNS can assemble a working, best-practice, validated solution in the blueprint. Notice that the connection between the blueprint and the infrastructure is a two-way arrow. The blueprint not only pushes configuration to the infrastructure, but also the infrastructure informs the blueprint about state changes and anything else that can effect intent compliance. In other words, the blueprint (context) contains the intent and operational state knowledge.

Pushing It All to the Infrastructure

The most efficient means for an IBNS to communicate with an infrastructure device is to install a software agent on the device that serves as the IBN local representative. Think of the software agent as the local manager that problem customers always want to see. But not all devices can accommodate an agent installation, or the IBN agent software may not support all devices. So it's important for the IBNS to be able to have multiple means of communication, shown in Figure 3-7, with infrastructure devices,

such as open or vendor specific APIs or by directly logging into the device via SSH.

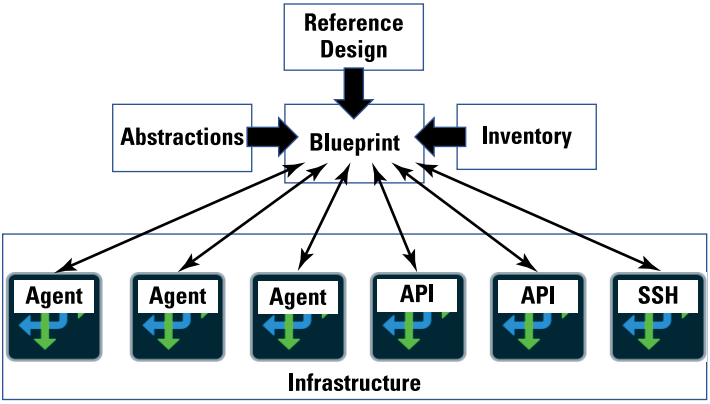

FIGURE 3-7: The IBNS must have multiple means of communication.

IN THIS CHAPTER

- Learning about change
- Taking action on your insights
- Examining an IBA probe
- Leveraging IBA and reference designs

Chapter 4
Staying Alert with Intent-Based Analytics

Chapter 3 shows you a basic intent-based networking (IBN) architecture that provides intent fulfillment. But fulfilling your expressed intent, all by itself, isn't good enough because networks change — both intentionally and unintentionally. A proper IBNS uses intent-based analytics (IBA) to stay aware of network changes in real time, continually ensuring that your services remain in compliance with your intent. That's *intent assurance*.

This chapter shows you why your IBN system (IBNS) requires sophisticated, deep analytics that can detect when a deployed service is drifting out of spec and either automatically make the adjustments to bring it back into compliance or alert you to the problem.

Experiencing Two Types of Change

Every network experiences both intentional and unintentional changes; that's easy enough to understand. But the consequences of those two types of change can be surprisingly convoluted.

Uncontrolled change

Things break. That's about the easiest way to define uncontrolled change. Your network is humming along, you haven't touched anything (as far as you know), and suddenly an interface goes down. Or a device crashes. Or a protocol stops working. Or buffers overflow. Or one of dozens or maybe hundreds of other possible failures occur. So many delightful possibilities!

Some failures, such as a broken link, present obvious symptoms and are easy to diagnose. Others, such as sub-optimal routing and intermittent packet drops, can be so subtle that they're difficult to detect, much less diagnose. Even more challenging is identifying indications of an impending failure before it happens.

When dealing with uncontrolled changes, three IBNS features are essential:

- Built-in analytics that can detect deviations from expressed intent
- Probes that pull essential data and state from the network
- Fast root cause analysis

Controlled change

Controlled change is change that you intend to happen. You change a configuration, or you add change or remove an element. Whether that change is *well* controlled is a different matter.

Pre-deployment verification checks in an IBNS certainly can help prevent problems with configuration changes, but you're still going to run into instances where you make a change and things break. Nobody is perfect — breathe and accept that fact. If the problem is serious, your first priority is quickly getting back to a known working configuration and then figuring out things from there. If the problem isn't immediately service impacting, you want your IBNS to help you sort through possibly massive telemetry data to pinpoint the source of the problem.

IBNS features that are essential to dealing with controlled changes include the following:

- Reliable pre- and post-deployment verification
- Fast, accurate rollback capability
- Flexible network probes
- Fast root cause analysis

Discovering Actionable Insights

Network operators take in vast amounts of network data and then struggle to extract meaningful insights from it all. In a "cloud" of thousands of network data points, such as the ones depicted in Figure 4-1, how do you decide which ones are important?

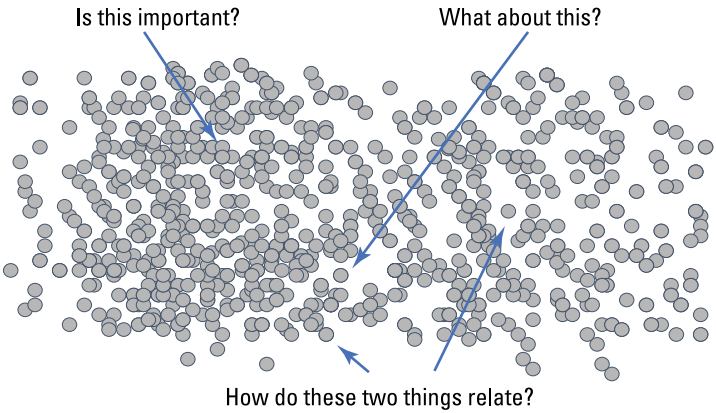

FIGURE 4-1: The dizzying confusion of data points.

They can't just clear the fog by reducing the volume of information. A data point that's irrelevant today may be crucial tomorrow. As a result, operators face an explosion of operational expenses when trying to make sense of their data fog and trying to extract up-to-the-second, relevant information.

The key to making sense of all the data you gather is to put it all into real-time, *queryable* context. There are two aspects to this:

- You must be able to detect conditions of interest. You need analytical probes that ask essential questions and root out important information, stripping away what's irrelevant.

» Conditions of interest, in turn, are classified by the relationships between conditions.

Relationships between data points give you a strong hint about how IBA works: The fog of data roughly depicted in Figure 4-1 is stored as a graph, where the relationships between data points are as essential as the points themselves. Within the context that you express, the relationships lead you to the answers you need, shown in Figure 4-2.

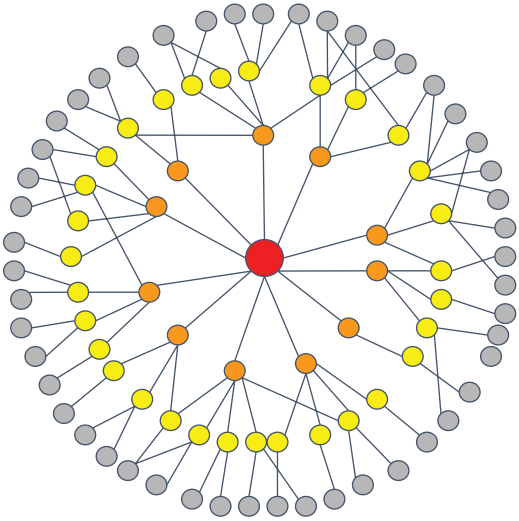

FIGURE 4-2: A graph data store leads you to the specific answers you need.

Probing Your Data

What information you draw from your graphical data store depends on the particular circumstances of your network and what's going on at a given slice of time. You may want to be alerted to intent drift, or you may be actively trying to identify the root cause of a problem. You need to be armed with a means of probing your data store.

A simple example of a probe is

1. **Collect information from X, Y, and Z.**
2. **Analyze that collected data to understand what's normal over time.**

3. **Create a baseline from that normalized data.**
4. **If telemetry shows a drift off of the baseline of more than this defined threshold, alert me.**

This simple process, shown in Figure 4-3, allows you to deploy probes that are meaningful to your network and your services. It also means that when things change, you can deploy probes in minutes to get to the root of the problem.

FIGURE 4-3: A probe can be represented over time or on the fly.

An IBNS should provide you with a library of prebuilt probes that you can use as-is or modify to your needs. But no set of prebuilt probes is going to answer every question you have about your particular network. A good IBNS enables you to define your own probes.

A library of prebuilt probes is essential to IBA but can't address every eventuality. It's essential that you also have the ability to define your own probes quickly and accurately.

Getting to the Root

Something goes wrong. Red lights start flashing. Klaxons sound. Smoke starts billowing out of consoles. Okay, that's just a scene from most *Star Trek* episodes. But when a serious network problem crops up, your operators can certainly start feeling like that's what's happening. A fault in the network causes a flood of anomalies, some directly related to the cause and some only peripheral to it. Being able to sort through this storm of anomalies and center on the root cause is invaluable to quickly resolving problems.

Making the identification of a root cause is complicated by the fact that the root cause itself isn't always observable. For example, a security team may make a change on a firewall that breaks a BGP session to an external router. The firewall is outside your administrative domain, and you can see the impact of the policy change, but you don't have visibility to the firewall itself.

In such a case, sorting through the observed symptoms to make inferences about the root cause can be highly complex without system intelligence and automation. *Root Cause Identification* (RCI) is built on IBA but also utilizes the reference design I detail in Chapter 3. While IBA is focused on identifying complex symptoms in your network, RCI focuses on the *causes* of those symptoms. The reasoning defined in the reference design — which is specific to a network service — differentiates between symptoms and anomalies and how they all relate to each other. The result should be a sorting of anomalies, such as shown in Figure 4-4, from low fidelity to high fidelity, that leads you back to a root cause.

FIGURE 4-4: RCI analyzes the relationships between anomalies in your network to trace back to a root cause.

Reaching Back in Time

Sometimes when things blow up unexpectedly as the result of a controlled change, your immediate priority is to get your network back to a previously good state and then analyze what went wrong. Plenty of vendors implement configuration rollback capabilities with varying levels of usefulness, but you should be able to roll back a multi-vendor network all at once without resorting to the capabilities and procedures of each vendor.

Fortunately, an IBN system is perfectly positioned to do just that. The architecture shown in Figure 4-5 can pull and store configurations from all network nodes at multiple times, giving you a stored configuration history similar to what Time Machine does in macOS. Should you need to "travel back in time," your IBNS allows you to select a point prior to a network change and push that configuration back to your network. You don't even need a DeLorean and 1.21 gigawatts.

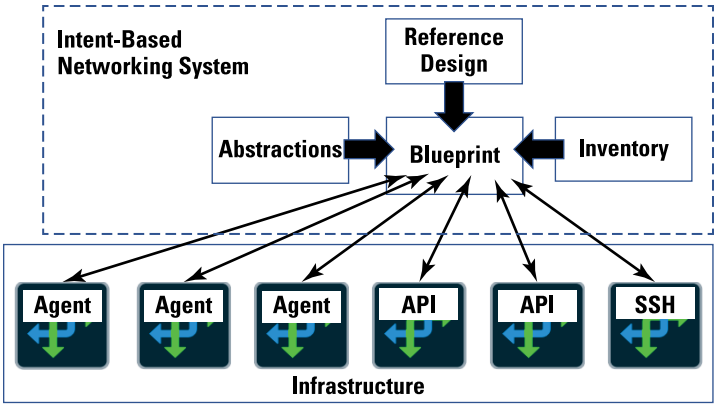

FIGURE 4-5: An IBNS can easily take vendor-neutral snapshots of previous configurations.

IN THIS CHAPTER

- Starting with intent
- Avoiding a vendor-first approach
- Steering clear of home-grown solutions
- Empowering new ways of thinking

Chapter 5
Ten Things to Think About When Considering IBN

Every *For Dummies* book has a Part of Tens chapter, and you've arrived at this one. Here, I give you ten considerations to keep in mind when evaluating intent-based networking (IBN) solutions. (Okay, I realize there are only seven items, but I don't make the chapter title rules.)

Don't Start with Hardware

Selecting a hardware vendor and then designing your network within the constraints of what that product can do isn't the smartest decision. Smart network and network service design have always started with intent. You stipulate your business intent, you design the services that fulfill that intent, and you design your network to support those services. Only then do you shop around for the vendors that have the capabilities that support the network you designed.

Whether you're designing a network from scratch or designing services for an existing network, IBN is an essential tool because it starts where you start — with business intent. As you work down to the specifics, you leverage the industry best practices integrated into reference designs, then network and service abstractions, and only then consult inventory to fit what you have or assess what's available to you.

Free Yourself from Vendor Lock-In

Avoiding a vendor-first approach goes beyond your initial designs and follows throughout your network life cycle. When you deploy new services or modify your network, you again want to start with your intent and let IBN take you to the best vendor (and vendor model) that supports your intent. Even if you're a single-vendor shop, IBN keeps you informed of all options available to you.

Sort through the Fluff

IBN is a hot marketing term because IBN is widely acknowledged as a major industry trend. Its benefits are well understood. I cover the benefits in Chapter 1.

WARNING

The downside of this is that most everyone claims to have an IBN solution. To meet those claims, the definition of IBN is often "massaged" to fit the realities of what someone is actually offering — including the limits of what's being offered. There's even a term for it — *intent washing* (and it's not all that clean).

Make sure you know what IBN really is, what its true benefits are, and most importantly, what you want to get out of it.

Don't Rely on Home-Grown Solutions

DevOps is a huge industry trend, and for good reason. Integrating development and operations removes traditional roadblocks to the efficient rollout of new applications and services. The trend has also pushed development skillsets as automation tools and

programming into operations. It's no longer good enough to be a "CLI jockey." Operators need some programming chops.

But there's also the danger of becoming overly reliant on operations tools that are built in-house. The tools might be built well-adapted to the specifics of your network, but can they change as your network changes? When a vendor changes its capabilities, what is the development effort involved in adapting your tools to those changes? And are you sure your in-house tools truly follow industry best practices?

Automation Is Only Part of the Story

Automation is fundamental to reducing *operational expenses* (OPEX) by shrinking operational tasks from weeks to minutes. But don't lump IBN in with automation tools because automation is only a part of the story. By definition, IBN translates expressed intent into verified configurations. Only then can automation, in the traditional sense, come out to play.

And all that is just the intent fulfillment aspect of IBN. Intent-based analysis (IBA) ensures that your network remains in compliance with your intent throughout the service life cycle, and root cause identification (RCI) provides rapid identification of root causes when things go wrong.

Rid Yourself of Calculated Guesswork

The bane of every network operator is sifting through overwhelming volumes of data to find meaningful information. This scenario is true whether you want to query your network during steady state, setting thresholds to alert you when state is drifting away from your expectations, or — and perhaps especially — when trying to get at the root of a problem.

WARNING

"Big data" telemetry without actionable insights can result in an explosion of operational expenses as you expend major resources to extract the data you need. IBN is an essential analytical tool for reducing your data analysis effort from hours, days, or weeks to minutes. Not only do you have far more efficient, economical, and agile network, but also you free up key resources for more important work.

Empower New Ways of Thinking

Top-tier architectural teams are often wasted performing tactical operations (commonly known as *firefighting*) instead of devoting their expertise to strategic planning. Frontline operations personnel can become bogged down in the minutiae of data searches, discrete element management, and change management instead of stepping back from the trees to see the complete network forest.

IBN provides the insights into your network services from Day 0 through Day 2, enabling your teams to think about your network as a complete service rather than a collection of nodes and links.

REMEMBER

Experience has demonstrated that IBN can

- Reduce time to delivery by 90 percent.
- Eliminate 85 percent of all outages by in-depth verification before deployment.
- Lower OPEX by up to 83 percent.
- Lower capital expenses (CAPEX) by up to 60 percent.
- Lower mean time to resolution by 70 percent.

Intent-Based Networks

A Revolutionary Approach to Data Center Automation

Overcome Resource & Staff Constraints

Improve Agility & Reliability

Increase Speed & Flexibility

apstra.com

IBN is the future of network operations

If you've heard of IBN, you may be confused by the publicity around it. Is it just marketing hype, or is there substance and value to IBN? This book establishes what *intent* means in an operational context and defines IBN in terms of its most important characteristics. You see a simple architecture for translating and fulfilling intent and how this architecture goes on to support expressed intent throughout a service life cycle and helps you find the information you need about your network, when you need it.

Inside...

- Discover the traits of an IBN solution
- Understand what IBN can do for you
- Look at the components of an IBN system
- Learn the value of intent-based analysis
- Find root causes in network anomalies
- Learn considerations in IBN deployment

Jeff Doyle is a Member of Technical Staff (MTS) at Apstra. A well-known author, Jeff has specialized in advanced networking technologies for most of his 30-year career.

ISBN: 978-1-119-68377-3
Not For Resale

for dummies®
A Wiley Brand

Also available as an e-book

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.