Russ White
Jeff Tantsura

Navigating Network Complexity

Next-generation Routing with SDN, Service Virtualization, and Service Chaining

FREE SAMPLE CHAPTER

Navigating Network Complexity

This page intentionally left blank

Navigating Network Complexity

Next-generation Routing with SDN, Service Virtualization, and Service Chaining

Russ White
Jeff Tantsura

Addison-Wesley
800 East 96th Street
Indianapolis, Indiana 46240 USA

Navigating Network Complexity

Next-generation Routing with SDN, Service Virtualization, and Service Chaining

Published by: Addison Wesley

Text printed in the United States on recycled paper at RR Donnelley, Crawfordsville, IN

First Printing November 2015

Library of Congress Cataloging-in-Publication Number: 2015950654

ISBN-13: 978-0-13-398935-9
ISBN-10: 0-13-398935-6

Publisher
Paul Boger

Associate Publisher
David Dusthimer

Executive Editor
Brett Bartow

Senior Development Editor
Christopher Cleveland

Managing Editor
Sandra Schroeder

Project Editor
Mandie Frank

Copy Editor
Cenveo® Publisher Services

Technical Editors
Ignas Bagdonas
Jon Mitchell

Editorial Assistant
Vanessa Evans

Cover Designer
Alan Clements

Composition
Cenveo® Publisher Services

Indexer
Cenveo® Publisher Services

About the Authors

Russ White began his network engineering career installing terminal emulation cards and inverse multiplexers in the United States Air Force. In 1996, he moved to Raleigh, N.C., to join Cisco Systems in the Technical Assistance Center (TAC) routing protocols team. From TAC, Russ moved to the global escalation team, and then into engineering, and finally into sales as a Distinguished Architect. He is currently a network architect working in the area of network complexity and large scale design, a member of the IETF Routing Area Directorate, an active speaker and writer, and active in the Internet Society. He holds CCIE #2637, CCDE 2007:001, the CCAr, a Masters in Information Technology from Capella University, and a Masters in Christian Ministry from Shepherds Theological Seminary. He lives in Oak Island, N.C., with his wife and two children, and is currently a P.h.D student at Southeastern Baptist Theological Seminary.

Jeff Tantsura started his network engineering career in early 1990s at a small ISP as system/network administrator, later working for bigger ISPs where he was responsible for network design and architecture, vendor selection.

Currently Jeff is heading Technology Strategy Routing at Ericsson as well as chairing IETF Routing Working Group.

Jeff holds MSc in Computer Science and Systems Analysis from University of Georgia and Executive Certificate of Business Excellence from Haas School of Business, Berkeley.

He also holds CCIE R&S #11416 and Ericsson Certified Expert IP Networking #8

Jeff lives in Palo Alto, CA, with his wife and youngest child.

About the Technical Reviewers

Ignas Bagdonas

Ignas Bagdonas has been involved in the network engineering field for last two decades, covering operations, deployment, design, architecture, development, and standardization aspects. He has worked on multiple large SP and enterprise networks worldwide, participated in many of the world's first technology deployments, and has been involved with building community awareness via conferences, seminars, and workshops. His current focus covers end-to-end network architecture evolution and new emerging technologies. Ignas holds Cisco CCDE and CCIE certifications.

Jon Mitchell

Jon Mitchell, CCIE No. 15953, is a network engineer in Google's Technical Infrastructure organization where he works on their global backbone. Prior to Google, Jon has worked in roles of network architecture at Microsoft, systems engineering at Cisco Systems, network architecture and engineering at AOL, and network engineering at Loudcloud for the last 15 years that he has been in the networking industry. He is also an active IETF participant in various routing area working groups. Through all of these roles, Jon has always had a passion for working on large-scale problems and solving them through simplification and automation. When Jon is not thinking about networking, he enjoys many other passions such as hiking, running, supporting clean water, microbreweries, and spending time with his wife and four children.

Dedications

Russ White: *This book is dedicated to Bekah and Hannah. Thank you for sticking with your grumpy old dad through thick and thin.*

Jeff Tantsura: *This book is dedicated to my family: Marina, Ilia, Miriam, and Davy—thank you for your support!*

Acknowledgments

Russ White:

I would like to thank the many people who have taught me networking through the years, including Denise Fishburne, Don Slice, Alvaro Retana, Robert Raszuk, and a host of others—too many to name. I would also like to thank Dr. Doug Bookman for driving me to be a better thinker, Dr. Will Coberly for driving me to be a better writer, and Dr. Larry Pettegrew for driving me just to be a better person. Finally, I'd like to thank Greg Ferro and Ethan Banks for inspiring me to start writing again.

Jeff Tantsura:

I would like to thank the people who taught and helped me through the years to understand networking better: Acee Lindem, Tony Przygienda, Tony Li, Jakob Heitz and many others, too many to help.

Special thanks to my co-author Russ for inspiring me!

Contents at a Glance

Contents

This page intentionally left blank

Introduction

Every engineer, no matter what type of engineering they do, face complexity almost constantly. The faster, cheaper, higher quality triad are the constant companion of everything the engineer does. Sometimes, though, the complexity isn't so obvious. For instance, how long will the software project take? Two hours, two weeks, or too long is the common reply—another encounter with complexity.

While research into complexity theory has proceeded apace in the scientific and mathematical worlds, the application of the theories and ideas to more practical engineering problems, particularly in a way that the "average engineer" can read and understand, simply hasn't kept pace. This book aims to fill that gap for network engineering.

In a move that will probably be disappointing to readers with a math degree, this book is devoid of the elegant mathematical models complexity theorists are working with. Rather, the focus is on the practical application of the theoretical constructs of complexity. Instead of being focused on the "pure theory" and math, this book is focused on the practical application of the ideas being investigated by complexity theorists.

This book, then, is targeted at the "average" network engineer who wants to gain an understanding of why particular common constructs work the way they do, such as hierarchical design, aggregation, and protocol layering. By getting behind these widely accepted ways of designing a network, exposing the reasons for the tradeoffs designed into the system in each case, the authors hope the reader learns to take lessons learned in one area and apply them to other areas. After reading this book, network engineers should begin to understand why hierarchical design and layering, for instance, work, and how to see and change the tradeoffs in more specific situations.

How This Book Is Organized

This book begins, in the first chapter, with a high level view of complexity theory. This is not deep theory, but it is rather designed to provide a hands-on view

of complexity without diving into heavy math (or any math at all). The second chapter provides an overview of various attempts at measuring complexity in a network, including some of the problems plauging each attempt. The third chapter proposes a model of complexity that will be used throughout the rest of the book.

Chapter 4 through 7 consider examples of complexity in network and protocol design. The point these chapters attempt to drive home is how the models and concepts in the first three chapters. Chapter 8 is something of an important detour into the world of complex system failure in network engineering terms. The rest of the book is dedicated to first providing an overview of three new technologies network engineers face at the time of writing, and then an analysis of each one in terms of complexity tradeoffs.

Note, Documents referenced with the form draft-xxx are IETF works in progress, and therefore do not have stable document names or uniform resource locators (URLs). Because of these limitations, explicit references have not been given, but rather just the title of the document and the document name. To find these documents , please perform a document search at the IETF website (ietf.org).

Chapter 3

Measuring Network Complexity

Given these four fundamental aspects of complexity—state, speed, surface, and optimization—it only makes sense to measure these four points and generate a single number describing the overall complexity of a given design and deployment structure. It would be nice if there were some way to examine a proposed network design, or a proposed change to a network design, and be able to assign actual numbers to the complexity of each component so the complexity can be compared to any potential gain in performance, or the loss of complexity in one area can be compared to the gain in complexity in another. If it were only that simple.

As it turns out, the effort to measure complexity is, itself, quite complex.

Two problems rise to the surface when examining the problem of measuring and quantifying a network toward gaining an understanding of the overall system complexity. *First,* there is the sheer amount of information available. Given the current push toward big data analytics, and the ability to measure thousands to millions of interactions and data mining to discover important trends and artifacts, shouldn't something the size of an average network be an easy problem? Consider some of the various points of measurement just in trying to understand the interaction between the data flowing through each point in the network and the queuing mechanisms used to handle that traffic. This might include things such as:

- The amount of data flowing through each point in the network, including the input and output queue of each forwarding device.
- The depth and state of each queue of each forwarding device in the network.

- The source, destination, and other header information of each packet forwarded through the network.
- The number of packets dropped by each forwarding device, including the reason why they were dropped (tail drop, packet error, filtered, filtering rule, etc.).

Considering that the measurements themselves must pass through the network—and the measurements can easily contain more traffic than the measured traffic—the problems with measuring everything should quickly become apparent. How can you separate the measurement from the measured if the measurement is being carried on the same channel as what you are measuring? Added to this challenge are the states of each individual control plane system, and the components of those systems—things like the memory and processor utilization of each forwarding device, the state of each adjacency between each pair of devices participating in the control plane, and the flow of each reachability advertisement within the control plane. To make measuring the system complexity even more complex, the interactions between the systems must also somehow be taken into account—things like the impact of reachability information on the distribution and application of policy, any interdependencies between parallel control planes in terms of reachability information and system resources, and interactions between overlay and underlay control planes. Measuring not only the systems but also the interactions between the systems quickly becomes an intractable problem.

When measuring a system to understand its complexity level, some sort of sampling must take place. Sampling necessarily means that some information must be left out—which, in turn, means that any measurement of complexity along these lines is necessarily an abstract representation of the complexity, rather than a measure of the complexity itself.

To top all of this complexity off, there is very little agreement on the set of things to measure to create even an accurate abstract representation of the complexity of a network.

There is a second problem looming on the horizon just past this first one—a problem that's not so obvious, and actually makes the problem of measuring network complexity intractable. Network design represents ordered (or intentional or organized—these three terms are often used interchangeably) complexity, rather than unordered complexity. While data analytics deals with unordered data well enough, ordered complexity is an entirely different problem set.

Let's begin by examining some methods proposed to measure network complexity, and then consider ordered versus unordered complexity. Finally, several realms of complexity will be examined that will lead to practical applications.

Some Measures of Network Complexity

The difficulty of the task hasn't stopped researchers from attempting to measure network complexity. Quite the opposite—there are a number of methods that have been tried over the years. Each of these methods has contributed useful thinking to the problem space, and can actually be used to provide some insight into what network complexity looks like. Overall, though, none of these measurements will truly provide a complete view of the complexity of a network.

Let's look at three examples of network complexity measurements to get a feel for the space.

Network Complexity Index

The Network Complexity Index is described in "A Network Complexity Index for Networks of Networks"[1] by Bailey and Grossman (commonly called the NCI). The general idea is to tackle describing network complexity in two steps:

- Break the network down into subnetworks. As described in the original paper:

  Given a network N, we first divide the network into smaller sub-networks C[1], . . ., C[j], . . ., C[p] with the property that two nodes selected at random from the sub-network C[i] are more likely to be connected to each other than two nodes selected at random from outside the sub-network (N\C).

- Compute the complexity based on the size and number of the subnetworks. Again, as described in the original paper:

  Given the sub-communities of the network N, let X[j] denote the size of the j largest sub-community, so that the sequence X[1], . . . , X[p] is in decreasing order. In general, different communities may have the same size. We define the network complexity index B(N) of the network N as the solution of the following equation: B(N) = Max j, X[j] j

The equation given is a standard statistic used in evaluating the importance of scientific research known as the H-index. The H-index determines the impact of

1. Stewart Bailey and Robert L. Grossman, "A Network Complexity Index for Networks of Networks" (Infoblox, 2013), n.p., https://web.archive.org/web/20131001093751/http://flowforwarding.org/docs/Bailey%20-%20Grossman%20article%20on%20network%20complexity.pdf.

a particular piece of research by evaluating the number of citations of the work in a way that is similar to a web page search index using the number of links to a page to determine the importance or relevance of that page.

Seen this way, the NCI attempts to combine the connectivity within a network with the number of nodes within a network:

- The more the subcommunities, the more connection points there must be between these subcommunities, and hence the more complex the connection graph must be.
- The larger the subcommunities, the more nodes there are within the network; this again impacts the implied connectivity graph of the network

The size and scope of the connectivity graph, in turn, impacts the way information flows within the network, which also relates to the complexity of the network.

What the NCI Does Well

The NCI does a good job of producing a single number that describes the size and shape of a network in terms of nodes and communities, in turn implying the scope and complexity of the network interconnections. This single number, computed over time, can help network managers and designers understand the growth of a network in terms other than sheer size.

What the NCI Doesn't Do

From a network engineer's perspective, there are several practical problems with using the NCI as a single measure of network complexity. *First,* this isn't something you're going to compute on a napkin while you're eating dinner, or do rough calculations in your head around.[2] This is a math heavy computation that requires automated tools to compute. *Second,* other than measuring the growth and interconnectedness of a topology, it's hard to see how and where the NCI is useful in the real world. There's no obvious way to reduce network complexity as measured by the NCI other than reducing the number and size of the subcommunities in the network.

This second objection, however, leads to another shortcoming of the NCI: it doesn't really measure the complexity network operators interact with. It's quite

2. In fact, an entire project called Tapestry was built around measuring the NCI by gathering configurations automatically and running them through a processor. The project can be found on GitHub at https://github.com/FlowForwarding/tapestry.

common, in the real world, to find very large networks supporting only a few workloads that have been heavily optimized for that workload, and hence are not very complex from an engineer's point of view. It's also quite common, in the real world, to find small networks with a very diverse workload, and hence cannot be optimized for a single workload. These networks are more complex than their size indicates—the NCI would likely underestimate the complexity of these networks.

So what does the NCI miss? Just those pieces of network architecture that designers deal with most of the time, such as:

- Policy, expressed through configuration, metrics, protocols, and other methods
- Resilience, expressed through the amount of redundancy, fast convergence mechanisms, and other highly complex design components

So while the NCI is useful, it doesn't capture all the complexity of a single network in a way that can be usefully applied to real-world networks.

Modeling Design Complexity

In a set of slides presented to the Internet Research Task Force's Network Complexity Research Group, a group of researchers described a model for measuring and describing the complexity of enterprise routing design.[3] The process of measurement is as follows:

1. Decompose the network design into individual pieces, implemented as individual configuration components (across all devices in the network).
2. Build a network of connections between these individual components.
3. Measure this network to determine the complexity of the configuration.

Figure 3.1 is taken from these slides,[4] illustrating the linkages between the various configuration components.

3. Xin Sun, Sanjay G. Rao, and G.Xie Geoffrey, "Modeling Complexity of Enterprise Routing Design" (IRTF NCRG, November 5, 2012), n.p., accessed October 5, 2014, http://www.ietf.org/proceedings/85/slides/slides-85-ncrg-0.pdf.
4. Ibid.

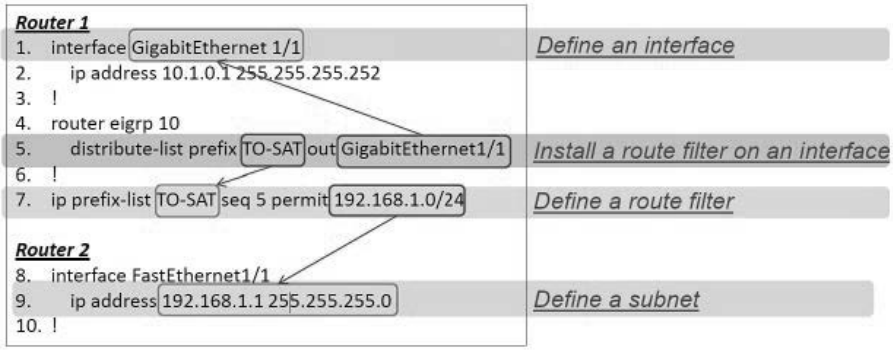


Figure 3.1 *Evaluating Linkages in a Network Configuration*

The more items configured, and the more dense the interconnection between the configurations (particularly between boxes), the more complex the design is determined to be. By examining these factors, the process yields a single number describing the complexity of the design. The presentation and the papers written by the same authors extend this concept to determining if the implemented design matches the design intent, given the design intent is stated in a way amenable to the process.

What Modeling Design Complexity Does Well

The concept of measuring not just the lines of configuration, but the interconnections between the lines of configuration, is extremely powerful. There is little doubt that much of complexity comes from the interrelated configurations spread throughout a network required to implement a single policy or to make a single protocol or process work. Looking at the interactions, or network of configurations, also takes the number of lines of configuration somewhat out of the picture as a measure of complexity. Because some devices can express a policy in a few simple lines, while others require a lot of configuration to express the same policy, this is a useful outcome.

What Modeling Design Complexity Doesn't Do

At the same time, however, the interconnections between lines of configuration can fall prey to the same problems just counting the number of lines of configuration can fall prey to—an entire policy might be represented by a single line of configuration on one device, while requiring a number of lines of policy on another. For instance, on Cisco IOS Software, the command *remove-private-as* is used to remove any private AS numbers in a BGP route advertisement. This

single command essentially replaces a set of configuration commands that would necessarily be interconnected, such as a filter and an application of that filter to a particular set of BGP peers. Both configurations are valid and perform the same set of actions, but they would appear to have completely different complexity levels according to the measure as it's described. Further complicating the situation, different BGP implementations might use different sets of command to perform the same action, making one configuration appear more complex than another, although they're both implementing the same policy.

Another failing of the measure described above is that it's not always obvious what pieces fit together to make a policy. For instance, a configuration removing private AS numbers on every eBGP speaker in an AS might not appear to be related within the measurement; there is no specific point at which these multiple configurations overlap or interact in a way that's obvious, *unless you know the intent of the configuration*. Thus some policies might easily be missed as they consist of configurations with no obvious point at which they tie together.

Finally, it's difficult to assess how a single configuration used to implement multiple policies would be managed in this measure of network complexity—and yet, this is one of the thorniest problems to manage from a complexity standpoint, as this is precisely one of those difficult to manage interaction surfaces between otherwise unrelated policy implementations. How do the various policies measured interact? On this point, modeling design complexity is silent.

NetComplex

As previously discussed, the NCI measures complexity based on scale and perceived subcomponents; modeling design rates complexity on the network of interconnected lines of configuration. What about measuring complexity based on the amount of work needed to keep the distributed database that represents the network topology synchronized across all the devices participating in the control plane? This is precisely what NetComplex does. As Chun, Ratnasamy, and Kohler state:

> *We conjecture that the complexity particular to networked systems arises from the need to ensure state is kept in sync with its distributed dependencies. The metric we develop in this paper reflects this viewpoint and we illustrate several systems for which this dependency centric approach appears to appropriately reflect system complexity.*[5]

5. Byung-Gon Chun, Sylvia Ratnasamy, and Eddie Kohler, "NetComplex: A Complexity Metric for Networked System Designs" (5th Usenix Symposium on Networked Systems Design and Implementation NSDI 2008, April 2008), n.p., accessed October 5, 2014, http://berkeley.intel-research.net/sylvia/netcomp.pdf.

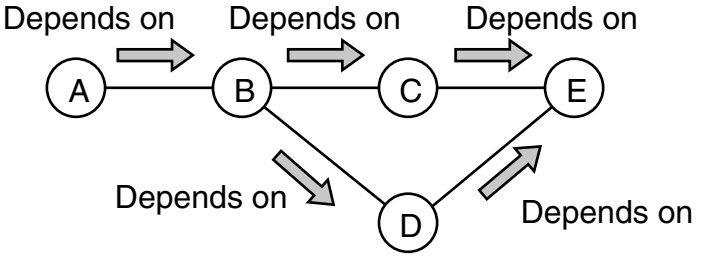


Figure 3.2 *Dependency and Complexity in NetComplex*

NetComplex evaluates the chain of dependent states in the network, assigns a metric to each dependency, and then calculates a single complexity measure based on these assigned metrics. Figure 3.2 illustrates dependency and complexity in NetComplex.

In this figure:

- Routers C and D depend on E to obtain a correct view of the network beyond Router E.
- Router B depends on Routers C, D, and E to obtain a correct view of the network beyond Router E.
- Router A depends on Routers B, C, D, and E to obtain a correct view of the network beyond Router E.

Hence, Router A "accumulates" the complexity of synchronization of information originating beyond E through the entire network. Through these dependencies, Router A is said to be linked to the remaining routers in the network. By examining these links, and combining them with the local state, the complexity of keeping the entire control plane synchronized can be given a single metric.

What NetComplex Does Well

By focusing on the amount of state and the way state is carried through the network, NetComplex does a good job of describing the complexity of a control plane. Based on this, NetComplex is useful for determining the additional complexity required to carry source routing information through the network, and forward based on this source routing information. Another place where Netcomplex would be useful is in putting a metric on the additional state information required to forward traffic on a per flow, rather than per destination/virtual topology basis.

What NetComplex Doesn't Do

NetComplex, however, is focused on the control plane within a single administrative or failure domain. There is no way, for instance, to account for the information hidden through route aggregation, nor to differentiate between topology information and reachability (such as what happens at a link state flooding domain boundary). NetComplex doesn't work with policies, nor policy implementation; nor does it deal with traffic flows, subnetwork scale, or network density.

Organized Complexity

Three different measures of network complexity have been examined at this point: NCI, modeling design complexity, and NetComplex. Each of these attempts to measure, in some way, at least some component of the four realms of network complexity—state, speed, surface, and optimization. None of them, however, measure everything in any one of these three domains, and none of them even come close to measuring overall network complexity. Why? The problem isn't just the ability to measure and process all the information needed to produce a single complexity number, it's embedded in the problem of network complexity itself.

Imagine, for a moment, a pool table with a set of balls on it. These specific balls are (at least nearly) perfect in their resilience, so they lose only infinitely small amounts of energy when they strike another object, and the bumpers on the sides of the table are designed in much the same way. There are no pockets in this table, either, so there is no place for the balls to leave the table. Now, place the balls on the table in some random distribution, and then strike one so it starts a chain reaction. The result will be a statistically random set of movements, each ball moving about the table, striking another ball or a bumper, retaining most of its energy, and then moving in a straight line in some other direction.

This particular problem is ripe for statistical regression analysis, or any other form of analysis data science can provide. The data scientist can tell you, based on a set of derived formulas, how often one ball will strike another, how long the system will take to run out of energy, what patterns will form in the randomly moving balls at what time—and many other things. Data science excels at finding patterns in seemingly random bits of data. In fact, it is often found that the data set must be larger to make an accurate prediction; the larger the data set, the more accurate characterization of it can be made, and the more accurate the predictions about the state of that data at some specific point in the future will be.

But let's change the situation somewhat. Let's take the same pool table, the same balls—all the same physical conditions. Only this time, someone has preplanned the position and movement of every ball such that no two balls strike one another, even though they are all in motion. In fact, the movement of every ball is identical throughout the entire time the balls are in motion.

What can data science tell us about this particular situation? *Nothing.*

Simple observation can tell us which ball will be where at any point in time. Simple observation might even be able to provide a formula telling us where there will be clumps of balls on the table, or near misses. But statistical analysis cannot go much beyond a few simple facts here. What's more interesting is that statistical analysis cannot tell us what the point is in having these balls arranged just this way.

This is the problem of *organized complexity.*

As Warren Weaver noted in 1948:

> *This new method of dealing with disorganized complexity, so powerful an advance over the earlier two-variable methods, leaves a great field untouched. One is tempted to oversimplify, and say that scientific methodology went from one extreme to the other—from two variables to an astronomical number—and left untouched a great middle region. The importance of this middle region, moreover, does not depend primarily on the fact that the number of variables involved is moderate—large compared to two, but small compared to the number of atoms in a pinch of salt. The problems in this middle region, in fact, will often involve a considerable number of variables. The really important characteristic of the problems of this middle region, which science has as yet little explored or conquered, lies in the fact that these problems, as contrasted with the disorganized situations with which statistics can cope, show the essential feature of organization. In fact, one can refer to this group of problems as those of organized complexity.*[6]

This field of organized complexity exactly describes the situation engineers face in looking at computer networks. No matter what angle a computer network is approached from, the problem is both complex and organized.

- Protocols are designed with a specific set of goals in mind, a specific mindset about how the problems approached should be solved, and a set of tradeoffs between current optimal use, future flexibility, supportability, and ease of implementation.
- Applications that run on top of a network are designed with a specific set of goals in mind.

6. Warren Weaver, "Science and Complexity," *American Scientist* 36 (1948): 539.

- Control planes that provide the metadata that make a computer network work are designed with a specific set of goals in mind.
- Protocols that carry information through the network, at every level, are designed with a specific set of goals in mind.

No matter which system within computer network is considered—from protocols to design to applications to metadata—each one was designed with a specific set of goals, a specific mindset about how to solve the problems at hand, and a specific set of tradeoffs. Some of these might be implicit, rather than explicit, but they are, nonetheless, intentional goals or targets.

A network is not just a single system that exhibits organized complexity, but a lot of different interlocking systems, each of which exhibits organized complexity, and all of which combined exhibit a set of goals as well (perhaps a more ephemeral set of goals, such as "making the business grow," but a set of goals nonetheless).

A Philosophical Aside

Within the realm of philosophy, there are those who believe that there is no such thing as organized complexity. Instead, what appears to be organized complexity is simply the result of emergence within any physical system once it becomes complex enough—that organization is somehow "built in" to the natural order, or into the way matter itself is formed and interacts. This school of thought believes that any and all actions can be traced back to some physical cause (for instance, that humans do not actually make decisions as much as decisions happen to humans). Whatever the reader's stand on this topic (and it is outside the scope of this book to argue the philosophical questions here), the practical result, in terms of network architecture, is: it doesn't matter. Networks are designed by people to solve a particular set of problems; no matter what is "behind" these designs, we must, to understand computer networks and their designs, get to the "why." Why did someone design this in this particular way? Why did someone make that particular tradeoff?

Network complexity, then, cannot simply be measured, computed, and "solved," in the traditional sense. Even everything could be measured in a single network, and even if all the information gathered through such measurement could be processed in a way that made some sense, it would still not be possible to fully express the complexity of a computer network in all its myriad parts—in essence because there is no way to measure or express intent.

Is This a Waste of Time?

None of this means it is wasting time to attempt to measure network complexity. What it does mean, however, is that the problem must be approached with a large dose of humility. Engineers need to be very careful about understanding the tradeoffs being made in every part of the design, and very intentional in remembering that there are limits to accurately predicting the outcome of any particular design decision.

Instead of "giving up," engineers should do everything possible to understand the complexity, to contain it, to minimize it, and to make intelligent tradeoffs—but there isn't, and won't ever be a silver bullet for complexity. As explained in Chapter 1, "Defining Complexity," there are sets of three out of which only two can be chosen, and there are curves where increasing complexity in one axis to solve a particular problem actually causes problems in another axis.

Measuring and managing complexity is not wasting time unless you believe you can actually *solve the problem*—because the "problem" cannot be "solved."

A Final Thought

This investigation of complexity has so far concluded:

- Complexity is necessary to solve difficult problems, particularly in the area of robust design.
- Complexity beyond a certain level actually causes brittleness—robust yet fragile.
- Complexity is difficult (or perhaps impossible) to measure in any meaningful way at the systemic level.
- There are a number of classes of problems where it is impossible to resolve for more than two of three goals (such as fast, cheap, high quality).

Given this set of points, it might seem like this is the end of the road. Network engineers are reliant on something that cannot be effectively measured—and measurement is always the first step in controlling and managing a problem set. There will never, in the end, be a single number or formula that can describe network complexity. Should we simply put on our pirate hats and proclaim, "Abandon hope all ye who enter here"? Or is there some way out of this corner?

There is, in fact, a reasonable way to approach complexity in the real world. Rather than trying to find an absolute "measure of complexity," or find some algorithm that will "solve" complexity, it's possible to construct a heuristic, or a method of looking at the problem set that will enable a path to a solution. The heuristic, in this case, is a two-part process.

First, expose the complexity tradeoffs inherent in network design. Exposing these tradeoffs will help engineers make intelligent choices about what is being gained, and what is being lost, when choosing any particular solution to a particular problem set. Every problem cannot be solved equally well; any solution applied at one point will increase complexity somewhere else.

To put it in other terms, network *engineers need to learn to be intentional about complexity.*

The next chapter will begin looking at three specific realms of complexity—operational, design, and protocol. In each of these cases, several places where designers must make tradeoffs to illustrate the process of bringing complexity out into the open will be considered. The closer engineers get to making intentional decisions about complexity when designing and managing networks, the more likely meeting the real-world demands placed on networks will be possible.

This page intentionally left blank

Index

T

U

This page intentionally left blank

REGISTER YOUR PRODUCT at informit.com/register

Access Additional Benefits and SAVE 35% on Your Next Purchase

- Download available product updates.
- Access bonus material when applicable.
- Receive exclusive offers on new editions and related products. (Just check the box to hear from us when setting up your account.)
- Get a coupon for 35% for your next purchase, valid for 30 days. Your code will be available in your InformIT cart. (You will also find it in the Manage Codes section of your account page.)

Registration benefits vary by product. Benefits will be listed on your account page under Registered Products.

InformIT.com–The Trusted Technology Learning Source

InformIT is the online home of information technology brands at Pearson, the world's foremost education company. At InformIT.com you can

- Shop our books, eBooks, software, and video training.
- Take advantage of our special offers and promotions (informit.com/promotions).
- Sign up for special offers and content newsletters (informit.com/newsletters).
- Read free articles and blogs by information technology experts.
- Access thousands of free chapters and video lessons.

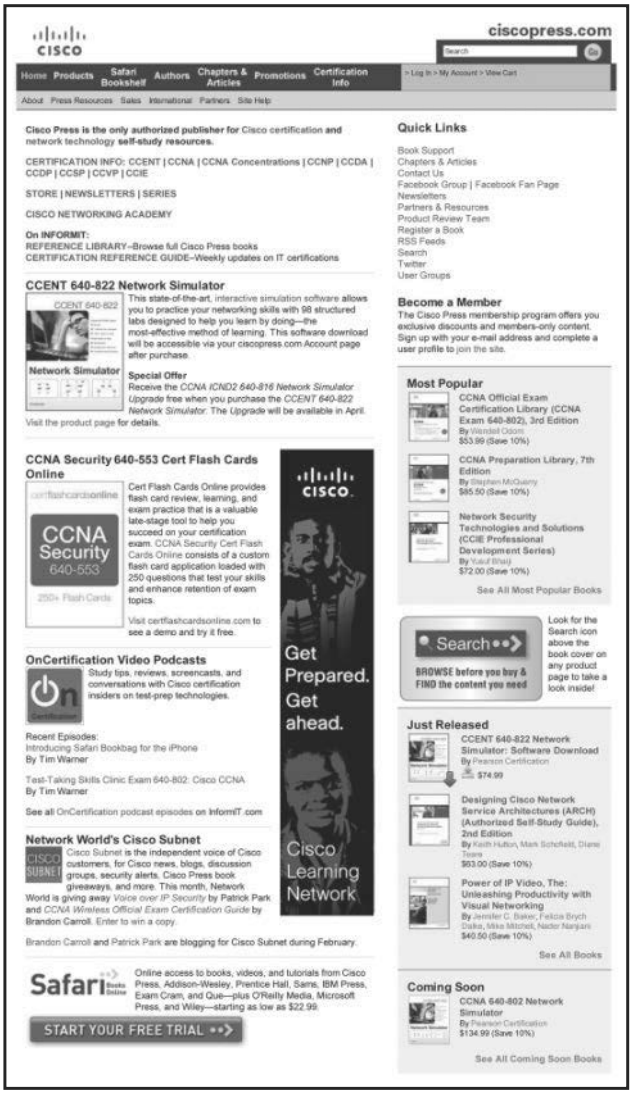
Connect with InformIT–Visit informit.com/community

Learn about InformIT community events and programs.

informIT.com
the trusted technology learning source

Addison-Wesley • Cisco Press • IBM Press • Microsoft Press • Pearson IT Certification • Prentice Hall • Que • Sams • VMware Press

ciscopress.com: Your Cisco Certification and Networking Learning Resource



Subscribe to the monthly Cisco Press newsletter to be the first to learn about new releases and special promotions.

Visit **ciscopress.com/newsletters.**

While you are visiting, check out the offerings available at your finger tips.

–Free Podcasts from experts:
- OnNetworking
- OnCertification
- OnSecurity

Podcasts

View them at **ciscopress.com/podcasts**.

–Read the latest author **articles** and **sample chapters** at **ciscopress.com/articles**.

–Bookmark the Certification Reference Guide available through our partner site at **informit.com/certguide**.

Connect with Cisco Press authors and editors via Facebook and Twitter, visit **informit.com/socialconnect**.

Connect, Engage, Collaborate

The Award Winning Cisco Support Community

Attend and Participate in Events

Ask the Experts
Live Webcasts

Knowledge Sharing

Documents
Blogs
Videos

Top Contributor Programs

Cisco Designated VIP
Hall of Fame
Spotlight Awards

Multi-Language Support

https://supportforums.cisco.com