Building a Future-Proof Cloud Infrastructure

A Unified Architecture for Network, Security, and Storage Services

SILVANO GAI

With Contributions by

Roger Andersson, Diego Crupnicoff, and **Vipin Jain**

Building a Future-Proof Cloud Infrastructure

A Unified Architecture for Network, Security, and Storage Services

Silvano Gai

With Contributions by
Roger Andersson,
Diego Crupnicoff, and Vipin Jain

Addison-Wesley

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

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

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

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2019956931

Pearson Education, Inc.

ISBN-13: 978-0-13-662409-7

ISBN-10: 0-13-662409-X

ScoutAutomatedPrintCode

Editor-in-Chief
Mark Taub

Product Manager
James Manly

Managing Editor
Sandra Schroeder

Senior Project Editor
Lori Lyons

Copy Editor
Paula Lowell

Production Manager
Vaishnavi/codeMantra

Indexer
Erika Millen

Proofreader
Abigail Manheim

Editorial Assistant
Cindy Teeters

Cover Designer
Chuti Prasertsith

Compositor
codeMantra

To the women in my family: Antonella, Eleonora,
Evelina, and Carola;
and to Jacopo

This page intentionally left blank

Contents at a Glance

	Preface	xix
Chapter 1	Introduction to Distributed Platforms	2
Chapter 2	Network Design	10
Chapter 3	Virtualization	34
Chapter 4	Network Virtualization Services	62
Chapter 5	Security Services	84
Chapter 6	Distributed Storage and RDMA Services	100
Chapter 7	CPUs and Domain-Specific Hardware	130
Chapter 8	NIC Evolution	142
Chapter 9	Implementing a DS Platform	156
Chapter 10	DSN Hardware Architectures	174
Chapter 11	The P4 Domain-Specific Language	190
Chapter 12	Management Architectures for DS Platforms	204
	Index	230

Contents

Preface xix

Chapter 1: Introduction to Distributed Platforms **2**

1.1 The Need for a Distributed Services Platform 3

1.2 The Precious CPU Cycles 4

1.3 The Case for Domain-Specific Hardware 4

1.4 Using Appliances 5

1.5 Attempts at Defining a Distributed Services Platform 6

1.6 Requirements for a Distributed Services Platform 7

1.7 Summary 9

Chapter 2: Network Design **10**

2.1 Bridging and Routing 11

2.1.1 L2 Forwarding 12

2.1.2 L3 Forwarding 12

2.1.3 LPM Forwarding in Hardware 13

2.1.4 VRF 14

2.2 Clos Topology 14

2.3 Overlays 16

2.3.1 IP in IP 18

2.3.2 GRE 18

2.3.3 Modern Encapsulations 19

2.3.4 VXLAN 19

2.3.5 MTU Considerations 22

2.4 Secure Tunnels 22

2.5 Where to Terminate the Encapsulation 23

2.6 Segment Routing 23

2.7 Using Discrete Appliance for Services 25
2.7.1 Tromboning with VXLAN 25
2.7.2 Tromboning with VRF 26
2.7.3 Hybrid Tromboning 27
2.8 Cache-Based Forwarding 27
2.9 Generic Forwarding Table 29
2.10 Summary 30
2.11 Bibliography 30
Chapter 3: Virtualization **34**
3.1 Virtualization and Clouds 35
3.2 Virtual Machines and Hypervisors 37
3.2.1 VMware ESXi 40
3.2.2 Hyper-V 41
3.2.3 QEMU 43
3.2.4 KVM 43
3.2.5 XEN 46
3.3 Containers 47
3.3.1 Docker and Friends 48
3.3.2 Kata Containers 49
3.3.3 Container Network Interface 49
3.3.4 Kubernetes 50
3.4 The Microservice Architecture 52
3.4.1 REST API 54
3.4.2 gRPC 54
3.5 OpenStack 55
3.6 NFV 57
3.7 Summary 58
3.8 Bibliography 58

Chapter 4: Network Virtualization Services **62**

4.1 Introduction to Networking Services 62

4.2 Software-Defined Networking 63

4.2.1 OpenFlow 64

4.2.2 SD-WAN 66

4.2.3 gRIBI 67

4.2.4 Data Plane Development Kit (DPDK) 68

4.3 Virtual Switches 69

4.3.1 Open vSwitch (OVS) 70

4.3.2 tc-flower 73

4.3.3 DPDK RTE Flow Filtering 74

4.3.4 VPP (Vector Packet Processing) 75

4.3.5 BPF and eBPF 76

4.3.6 XDP 76

4.3.7 Summary on Virtual Switches 78

4.4 Stateful NAT 79

4.5 Load Balancing 79

4.6 Troubleshooting and Telemetry 80

4.7 Summary 82

4.8 Bibliography 82

Chapter 5: Security Services **84**

5.1 Distributed Firewalls 85

5.2 Microsegmentation 86

5.3 TLS Everywhere 87

5.4 Symmetric Encryption 89

5.5 Asymmetric Encryption 89

5.6 Digital Certificates 90

5.7 Hashing 90

5.8 Secure Key Storage 90
5.9 PUF 91
5.10 TCP/TLS/HTTP Implementation 91
5.11 Secure Tunnels 92
5.11.1 IPsec 92
5.11.2 TLS 93
5.11.3 DTLS 94
5.12 VPNs 94
5.13 Secure Boot 97
5.14 Summary 97
5.15 Bibliography 98
Chapter 6: Distributed Storage and RDMA Services 100
6.1 RDMA and RoCE 103
6.1.1 RDMA Architecture Overview 106
6.1.2 RDMA Transport Services 108
6.1.3 RDMA Operations 108
6.1.4 RDMA Scalability 109
6.1.5 RoCE 109
6.1.6 RoCE vs iWARP 110
6.1.7 RDMA Deployments 110
6.1.8 RoCEv2 and Lossy Networks 112
6.1.9 Continued Evolution of RDMA 117
6.2 Storage 119
6.2.1 The Advent of SSDs 119
6.2.2 NVMe over Fabrics 120
6.2.3 Data Plane Model of Storage Protocols 120
6.2.4 Remote Storage Meets Virtualization 122
6.2.5 Distributed Storages Services 124
6.2.6 Storage Security 125

6.2.7 Storage Efficiency 125
6.2.8 Storage Reliability 126
6.2.9 Offloading and Distributing Storage Services 126
6.2.10 Persistent Memory as a New Storage Tier 127
6.3 Summary 128
6.4 Bibliography 128
Chapter 7: CPUs and Domain-Specific Hardware 130
7.1 42 Years of Microprocessor Trend Data 131
7.2 Moore's Law 132
7.3 Dennard Scaling 134
7.4 Amdahl's Law 135
7.5 Other Technical Factors 136
7.6 Putting It All Together 137
7.7 Is Moore's Law Dead or Not? 138
7.8 Domain-specific Hardware 139
7.9 Economics of the Server 139
7.10 Summary 140
7.11 Bibliography 140
Chapter 8: NIC Evolution 142
8.1 Understanding Server Buses 143
8.2 Comparing NIC Form Factors 144
8.2.1 PCI Plugin Cards 144
8.2.2 Proprietary Mezzanine Cards 146
8.2.3 OCP Mezzanine Cards 147
8.2.4 Lan On Motherboard 148
8.3 Looking at the NIC Evolution 149
8.4 Using Single Root Input/Output Virtualization 152
8.5 Using Virtual I/O 153

8.6 Defining “SmartNIC” 154
8.7 Summary 155
8.8 Bibliography 155
Chapter 9: Implementing a DS Platform **156**
9.1 Analyzing the Goals for a Distributed Services Platform 157
9.1.1 Services Everywhere 157
9.1.2 Scaling 157
9.1.3 Speed 158
9.1.4 Low Latency 158
9.1.5 Low Jitter 158
9.1.6 Minimal CPU Load 159
9.1.7 Observability and Troubleshooting Capability 159
9.1.8 Manageability 160
9.1.9 Host Mode versus Network Mode 160
9.1.10 PCIe Firewall 161
9.2 Understanding Constraints 161
9.2.1 Virtualized versus Bare-metal Servers 161
9.2.2 Greenfield versus Brownfield Deployment 162
9.2.3 The Drivers 162
9.2.4 PCIe-only Services 162
9.2.5 Power Budget 163
9.3 Determining the Target User 163
9.3.1 Enterprise Data Centers 163
9.3.2 Cloud Providers and Service Providers 164
9.4 Understanding DSN Implementations 164
9.4.1 DSN in Software 164
9.4.2 DSN Adapter 166
9.4.3 DSN Bump-in-the-Wire 168

9.4.4 DSN in Switch 169
9.4.5 DSNs in an Appliance....171
9.5 Summary....172
9.6 Bibliography....173
Chapter 10: DSN Hardware Architectures 174
10.1 The Main Building Blocks of a DSN....174
10.2 Identifying the Silicon Sweet Spot....176
10.2.1 The 16 nm Process 177
10.2.2 The 7 nm Process 178
10.3 Choosing an Architecture 178
10.4 Having a Sea of CPU Cores 179
10.5 Understanding Field-Programmable Gate Arrays 181
10.6 Using Application-Specific Integrated Circuits 183
10.7 Determining DSN Power Consumption 184
10.8 Determining Memory Needs....185
10.8.1 Host Memory....185
10.8.2 External DRAM....186
10.8.3 On-chip DRAM 186
10.8.4 Memory Bandwidth Requirements....186
10.9 Summary....187
10.10 Bibliography....187
Chapter 11: The P4 Domain-Specific Language 190
11.1 P4 Version 16....192
11.2 Using the P4 Language 193
11.3 Getting to Know the Portable Switch Architecture 194
11.4 Looking at a P4 Example....195
11.5 Implementing the P4Runtime API....199

11.6 Understanding the P4 INT 201
11.7 Extending P4 201
11.7.1 Portable NIC Architecture 201
11.7.2 Language Composability 201
11.7.3 Better Programming and Development Tools 202
11.8 Summary....202
11.9 Bibliography....203
Chapter 12: Management Architectures for DS Platforms 204
12.1 Architectural Traits of a Management Control Plane 205
12.2 Declarative Configuration 206
12.3 Building a Distributed Control Plane as a Cloud-Native Application....207
12.4 Monitoring and Troubleshooting....209
12.5 Securing the Management Control Plane....210
12.6 Ease of Deployment....211
12.7 Performance and Scale 212
12.8 Failure Handling....214
12.9 API Architecture....215
12.10 Federation....218
12.10.1 Scaling a Single SDSP 219
12.10.2 Distributed Multiple SDSPs....220
12.10.3 Federation of Multiple SDSPs 220
12.11 Scale and Performance Testing....223
12.12 Summary....227
12.13 Bibliography....227
Index 230

List of Figures

Figure 1-1 A Distributed Services Platform 3
Figure 1-2 Services 6
Figure 1-3 North-South vs. East-West 8
Figure 2-1 LPM Forwarding 13
Figure 2-2 A Clos Network 15
Figure 2-3 Customers and Infrastructure IP Addresses 16
Figure 2-4 Generic Encapsulation 17
Figure 2-5 IPv4 in IPv4 Encapsulation 18
Figure 2-6 GRE Encapsulation 19
Figure 2-7 VXLAN Encapsulation 20
Figure 2-8 VXLAN Encapsulation Details 21
Figure 2-9 VTEPs 21
Figure 2-10 Segment Routing Example 24
Figure 2-11 Example of Trombone at Layer 2 25
Figure 2-12 Example of Trombone at Layer 3 26
Figure 2-13 Cache-based Forwarding in HW 28
Figure 3-1 A Hybrid Cloud 36
Figure 3-2 Different Types of Hypervisors 38
Figure 3-3 A Virtual Switch 39
Figure 3-4 VMware ESXi 40
Figure 3-5 The Hyper-V Architecture 42
Figure 3-6 The KVM Architecture 43
Figure 3-7 Virtio 44
Figure 3-8 KVM, QEMU, Virtio 45
Figure 3-9 vSwitch with SR-IOV 46
Figure 3-10 The XEN Architecture 47
Figure 3-11 Classical Virtualization versus Container Virtualization 48
Figure 3-12 Kata Containers 49
Figure 3-13 Kubernetes Cluster Components 51

Figure 3-14 Microservices Deployed using Kubernetes 51
Figure 3-15 A Microservice Architecture 52
Figure 3-16 The OpenStack Architecture 56
Figure 3-17 NFV example 58
Figure 4-1 Main Components of an OpenFlow Switch 65
Figure 4-2 gRIBI 67
Figure 4-3 DPDK 69
Figure 4-4 OVS (Open Virtual Switch) 70
Figure 4-5 A Distributed OVS 71
Figure 4-6 The OVS Architecture 72
Figure 4-7 OVS in a NIC 73
Figure 4-8 OVS Offload with tc-flower 74
Figure 4-9 VPP 75
Figure 4-10 XDP 77
Figure 4-11 All Solutions in One Picture 78
Figure 4-12 Web Load Balancing 80
Figure 5-1 North-South vs. East-West 85
Figure 5-2 Microsegmentation 86
Figure 5-3 TLS Protocol Stack 88
Figure 5-4 IPsec Encapsulations 92
Figure 5-5 TLS Encapsulation 93
Figure 5-6 DTLS Encapsulation 94
Figure 5-7 Example of a VPN 95
Figure 6-1 Host Connected to Dedicated Networks 101
Figure 6-2 Hosts with Unified Networks 102
Figure 6-3 Unified Network Software Stack 102
Figure 6-4 InfiniBand Protocol Stack and Fabric Diagram 103
Figure 6-5 Cost of I/O, Kernel Network Stack versus Bypass 104
Figure 6-6 Kernel Buffer versus Zero-Copy 105
Figure 6-7 Software Stack versus Protocol Offload 105
Figure 6-8 One-sided Operations Ladder Diagram 106

Figure 6-9 QPs, WRs, CQs, and Scheduler/QoS Arbiter 107
Figure 6-10 RDMA Transport Services 108
Figure 6-11 RDMA over Ethernet Protocol Stack Evolution 110
Figure 6-12 RoCE versus iWARP 110
Figure 6-13 InfiniBand and Ethernet Shares 111
Figure 6-14 Priority Flow Control 113
Figure 6-15 DCQCN 114
Figure 6-16 One-way Latencies 115
Figure 6-17 Go Back N versus Selective Retransmission 116
Figure 6-18 Staging Buffer Cost of Selective Retransmission 117
Figure 6-19 Tail Drop Recovery 118
Figure 6-20 Remote SCSI Storage Protocols 119
Figure 6-21 NVMe-oF 121
Figure 6-22 Example of Storage Write 121
Figure 6-23 Push Model 122
Figure 6-24 Explicit Remote Storage 123
Figure 6-25 Hypervisor Emulates Local Disk Towards Guest OS 123
Figure 6-26 NIC-Based NVMe Emulation 124
Figure 6-27 Distributed Storage Services 127
Figure 7-1 42 Years of Microprocessor Trend Data 132
Figure 7-2 Transistor Count as a Function of Years 133
Figure 7-3 Intel Processor Transistor Count 134
Figure 7-4 Dennard Scaling 135
Figure 7-5 Amdahl's Law 136
Figure 7-6 Combined effect on single-thread performance 138
Figure 8-1 PCIe Root Complex 143
Figure 8-2 Intel I350-T2 Intel Ethernet Server Adapter 145
Figure 8-3 Broadcom NeXtreme E-Series 145
Figure 8-4 HPE Card in a Mezzanine Form Factor 146
Figure 8-5 OCP 3.0 Small and Large Cards 147
Figure 8-6 Mellanox ConnectX-5 in OCP 3.0 Form Factor 148
Figure 8-7 LOM on a Cisco UCS 149

Figure 8-8 SR-IOV Architecture 153
Figure 8-9 VirtIO Architecture 154
Figure 9-1 A Distributed Services Platform 156
Figure 9-2 Server with a Classical NIC 165
Figure 9-3 Server with a DSN-capable NIC 167
Figure 9-4 Bump-in-the-Wire 168
Figure 9-5 DSN in Switch 170
Figure 9-6 DSNs in an Appliance 171
Figure 10-1 Major Considerations in Silicon Design 176
Figure 10-2 Possible Architectures for DSNs 179
Figure 10-3 ARM Performance as a Packet Processor 180
Figure 10-4 High-Bandwidth Memory (HBM) 187
Figure 11-1 Example of P4 Version 14 Network Device 191
Figure 11-2 P4 Language Components 192
Figure 11-3 The PSA 194
Figure 11-4 The V1 Model Architecture 195
Figure 11-5 P4Runtime 200
Figure 12-1 Distributed Services Management Control Plane 205
Figure 12-2 Implementation of Declarative Configuration 207
Figure 12-3 Security Model of Management Control Plane 211
Figure 12-4 Functionality and Architecture of API Service 216
Figure 12-5 A Single DS Manager 219
Figure 12-6 Distribution of Multiple DS Managers 220
Figure 12-7 Federation of Multiple DS Managers 221

Figure Credits

Figure 2-8: Cisco Nexus 9000 Series Switches © Cisco system, Inc

Figure 3-5: Hyper V Architecture © Microsoft corporation

Figure 3-7: Virtio: An I/O virtualization framework for Linux by M. Jones © IBM Corporation

Figure 3-10: Xen arc diagram © The Linux Foundation

Figure 3-12: Kata Containers: Secure, Lightweight Virtual Machines for Container Environments by Scott M. Fulton, III © The New Stack

Figure 4-1: OpenFlow Switch Specification © The Open Networking Foundation

Figure 4-2: gRIBI © 2020 GitHub, Inc, https://github.com/opencon/gribi

Figure 4-3: DPDK © DPDK Project

Figure 4-4: Linux Foundation Collaborative Projects, "OVS: Open vSwitch," © 2016 A Linux Foundation Collaborative Project. http:// www.openvswitch.org/

Figure 4-5: Open v Switch © 2016 A Linux Foundation Collaborative Project

Figure 4-9: The Linux Foundation Projects, 'Vector Packet Processing (VPP)' © Cisco Systems, Inc.

Figure 6-11: Supplement to InfiniBandTM Architecture Specification, Volume 1, Release 1.2.1, Annex A17, RoCEv2, © 2010 by InfiniBandTM Trade Association. https://cw.infinibandta.org/document/dl/7781

Figure 6-12: © Copyright 2017. Mellanox Technologies

Figure 6-13: Top 500, Development over time, © 1993-2019 TOP500.org. https://www.top500.org/statistics/overtime/

Figure 7-1: Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten New plot and data collected for 2010–2017 by K. Rupp – Creative Commons Attribution 4.0 International Public License

Figure 7-2: Moore's law © Our World In Data

Figure 8-1: PCI Express® Basics & Background by Richard Solomon © PCI-SIG

Table 8-1: Specifications of PCI-SIG © PCI-SIG

Figure 8-4: Used with permission from Hewlett Packard Enterprise

Figure 8-5: Used with permission from Open Compute Project Foundation

Figure 8-6: Used with permission from Open Compute Project Foundation

Figure 8-9: Virtio: An I/O virtualization framework for Linux by M. Jones © IBM Corporation

Figure 10-3: Used with permission from Pensando

Figure 11-1: P4 Language Tutorial © P4.org

Figure 11-2: P4 Language Tutorial © P4.org

Figure 11-3: P416 Portable Switch Architecture (PSA) © P4.org

Figure 11-4: P4 Language Tutorial © P4.org

Figure 11-5: P4Runtime Specification © P4.org

Preface

The Motivation for Writing this Book

I decided to write this book after the success of the *Cisco Unified Computing System (UCS) Data Center* book, which I wrote with Roger Andersson and Tommi Salli.

I attribute the excellent response to that book to the significant amount of material that we included on protocols, technologies, and system evolution. We discussed several significant changes that were happening in data centers, from the blade server architecture to the dominance of Ethernet, to the trend toward I/O consolidation. We also made some predictions on the long-term evolution, some of which turned out to be right, while others didn't.

In the eight years that followed, the pace of changes has dramatically increased, and I thought it was time to provide an updated view of the technology trends as independently as possible from actual products.

This book focuses on core services like segment routing, NAT, firewall, microsegmentation, load balancing, SSL/TLS termination, VPNs, RDMA, storage, and storage services such as compression and encryption. These services are vital components whenever multiple users share any cloud architecture, be it private, public, or hybrid. In particular, this book is about distributed services platforms that can be implemented with multiple service modules located in different hardware components, such as NICs, appliances, or switches. Distributing these service modules as closely as possible to the final applications enables very high performance, low latency, low jitter, deep observability, and rapid troubleshooting.

Who Should Read This Book

This book targets all IT professionals who want to learn about the evolution of service architectures. In particular, this book is helpful to:

- Network engineers, for the L2, L3 forwarding, Clos networks, VLAN, VXLAN, VPN, and network services
- Cloud engineers, for multi-tenancy, overlay networks, virtual switching, and GFT
- Security experts, for firewalls, encryption, key management, and zero trust
- Application engineers, for load balancing, virtualization, and microservice architecture
- High-performance computing engineers, for RDMA applications
- Storage engineers, for NVMe and NVMe-oF, compression, deduplication, and encryption in motion and at rest

After reading this book, the reader will comprehend the problems with the current reality of discrete centralized appliances and will understand the need to migrate services as closely as possible to the applications. They will then have the right knowledge to evaluate different commercial solutions, compare them by asking pertinent questions, and make the right selection for her/his business.

Chapter Organization

Chapter 1, "Introduction to Distributed Platform," introduces the need for a Distributed Services Platform that offers superior security, cloudlike scale, hardware performance, and low latency and yet be software programmable. A platform that is easy to manage, operate, troubleshoot, and works for bare metal, virtual machine, and container workloads. The chapter explores the need for domain-specific hardware, programmable through a domain-specific language for network, security, and storage services, to satisfy the insatiable demand for processing, as Moore's law hits the limits of physics.

Chapter 2, "Network Design," presents standard network designs for clouds and enterprise data centers, reviewing L2/L3 forwarding algorithms used both in classical switch routers and in hypervisors, the requirements posed by Clos networks with their leaf and spine architecture, the role of overlays and how to secure them, segment routing, and the need for "tromboning" in the presence of discrete appliances.

Chapter 3, "Virtualization," is about the trend in public, private, and hybrid clouds and virtualization; it covers the differences between bare metal, virtual machines, and containers. It describes virtualization solutions like VMware and KVM, with particular emphasis on their network and service implications. It introduces the microservice architecture and container technologies as a possible implementation, with examples on Docker and Kubernetes. It concludes with examples of OpenStack and NFV.

Chapter 4, "Network Virtualization Services," introduces networking virtualization services, starting with SDN and OpenFlow trends and more recent efforts like gRIBI. It discusses DPDK, virtual switches and OVS, offloading techniques like tc-flower, DPDK RTE flow, eBPF, and VPP and tries to provide a taxonomy of these many efforts. Popular services like load balancing and NAT are also presented. The chapter ends with a discussion about telemetry.

Chapter 5, "Security Services," introduces security services. Starting with popular services like firewall, the discussion evolves into microsegmentation, followed by a deep dive on security with symmetric and asymmetric encryption, key storage, unique key generation, digital certificates, hashing, TLS/TCP implementations, and VPNs.

Chapter 6, "Distributed Storage and RDMA Services," presents RDMA and storage services. RDMA was born in the world of high-performance computing, but it is now used also in enterprise and cloud networks. NVMe is the new storage standard that replaces SCSI for high-performance storage

drives. NVMe is also overlaid on RDMA to provide networkwide access to storage resources in an arrangement called NVMe over Fabrics (NVMe-oF). NVMe-oF can also use TCP as an alternative to RDMA. This chapter describes this landscape with the addition of essential storage services such as encryption of data at rest, compression, and data deduplication.

Chapter 7, "CPUs and Domain-Specific Hardware," discusses servers used in clouds and data centers, looking in particular to the dramatic reduction in performance growth in recent years that calls for domain-specific hardware to take over service functions. Moore's law, Dennard scaling, Amdahl's law, and 42 years of microprocessor data are analyzed to understand the economics of the server better and to help the reader decide how to partition the server functions.

Chapter 8, "NIC Evolution," describes the evolution of network interface cards (NICs) from simple devices in charge of sending and receiving one packet at a time, to more complex and sophisticated entities capable of effectively supporting multicore CPUs, multiple types of traffic, stateless offloads, SR-IOV, and advanced parsing/classification capabilities. The term SmartNICs has recently appeared to indicate NICs that incorporate even more processing power to enable offload of networking processing from the host CPU. Understanding this evolution is essential because the NIC represents one of the possible footprints for a distributed services platform.

Chapter 9, "Implementing a DS Platform," introduces distributed services platforms, outlining their goals, constraints, and implementation. Obtaining a standard set of functionalities with a granular distribution is key to scaling the architecture to a vast number of users while creating scalability and maintaining low latency, low jitter, and minimum CPU load. The chapter compares possible implementations and trade-offs in greenfield and brownfield deployment scenarios.

Chapter 10, "DSN Hardware Architectures," describes possible hardware implementations of these Distributed Services Platforms, considering three main approaches: sea of processors, field programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs). It compares the advantages and disadvantages of each approach and draws some conclusions.

Chapter 11, "The P4 Domain-Specific Language," presents the P4 architecture that makes it possible to implement ASICs that are data plane–programmable at runtime, an important feature that marries the programmability of devices like FPGAs with the performance and power-saving capability of ASICs. The chapter ends with an analysis of future directions to make P4 more usable and flexible.

Chapter 12, "Management Architectures for DS Platforms," discusses the architectural components and design choices to build a modern management infrastructure, leveraging the concepts of distributed systems, stateless microservices, and API-driven software. It presents design trade-offs for building secure, highly available, high performance, and scalable software. It further discusses the practical aspects of a management system, like ease of deployment, troubleshooting, diagnosing, and integration with existing software ecosystems. Finally, the chapter touches upon federating the declarative intent across multiple clusters.

Help Improve This Book

If you uncover any remaining resilient bugs, please contact the authors by email at dsplatforms@ip6.com. We welcome general comments to the text and invite you to send them by email also.

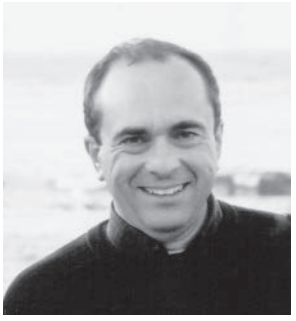
I hope this book provides useful information that you can apply to your daily activity.

—Silvano Gai

With Contributions by

- Diego Crupnicoff contributed all of Chapter 6, "Distributed Storage and RDMA Services" and helped review the book.
- Vipin Jain and Roger Andersson contributed Chapter 12, "Management Architectures for DS Platforms" and helped review the book.
- Francis Matus provided part of Chapter 7, "CPUs and Domain-Specific Hardware" on the historical evolution of processors as well as data for Chapter 10, "DSN Hardware Architectures."

About the Authors



Silvano Gai, who grew up in a small village near Asti, Italy, has more than 35 years of experience in computer engineering and computer networks. He is the author of several books and technical publications on computer networking as well as multiple Internet Drafts and RFCs. He is responsible for 50 issued patents. His background includes seven years as a full professor of Computer Engineering, tenure track, at Politecnico di Torino, Italy, and seven years as a researcher at the CNR (Italian National Council for Scientific Research). For the past 20 years, he has been in Silicon Valley where, in the position of Cisco Fellow, he was an architect of the Cisco Catalyst family of network switches, of the Cisco MDS family of storage networking switches, of the Nexus family of data center switches, and the Cisco Unified Computing System (UCS). Silvano is currently a Fellow with Pensando Systems.

Roger Andersson has spent more than 28 years in the computer and storage industry with work experience that spans across EMC/Data General, Pure Storage, Veritas/Symantec and Nuova Systems/Cisco UCS, focusing on software automation, OS provisioning, and policy-driven management at scale. Roger's roles started in hardware engineering, moved to software engineering, and for the past 16 years has been in technical product management roles. Roger is currently working at Pensando as a Technical Product Manager focusing on Distributed Service Management at scale. Roger was born in Stockholm, Sweden.

Diego Crupnicoff has been a Fellow at Pensando Systems since May 2017. Prior to that, Diego served as VP Architecture at Mellanox Technologies where he worked since its inception in 1999, driving chip and system architectures for multiple generations of Ethernet and RDMA products. Diego has been a member of the InfiniBand Trade Association since its early days and took part in the definition of the InfiniBand RDMA Standard. Among other roles, Diego chaired the IBTA Technical Working Group for many years. He was also among the founding directors of the OpenFabrics Alliance and chaired its Technical Advisory Council for several years. Over the past two decades, Diego has participated in multiple other SDOs and Tech Committees, including the IEEE802, IETF, T11, NVME, and the ONF. Diego is an inventor in multiple patents on the areas of computer networks and system architecture. He holds a B.Sc. in Computer Engineering (Summa Cum Laude) and an M.Sc. in EE. (Summa Cum Laude), both from the Technion - Israel Institute of Technology.

Vipin Jain is a passionate engineer with 20 years of industry experience. Over the years, he has contributed in the areas of switching, routing, network protocols, embedded systems, ASIC architecture, data path design, distributed systems, software-defined networking, container networking, orchestration systems, application security, open source, cloud infrastructure, and DevOps.

He holds numerous patents and has been a speaker at many conferences, author of IETF RFCs, and been a developer evangelist for his open source work. He enjoys coding for work and for fun. He also enjoys snowboarding, hiking, kayaking, and reading philosophy. He holds a bachelor's degree in Computer Science from NIT Warangal, India. He has worked in multiple successful startups in technical and management leadership roles. He is founder and CTO at Pensando Systems.

Acknowledgments

I would like to thank the following people for their contributions to this book:

- I am in debt to the MPLS team (Mario Mazzola, Prem Jain, Luca Cafiero, and Soni Jiandani) and to Randy Pond for having let me participate in the design of some of the most fantastic networking products of the last 20 years.
- John Evans and Boris Shpolyansky have spent many hours reviewing the chapters and providing countless insights.
- Rami Siadous, Bob Doud, Stuart Stammers, Ravindra Venkataramaiah, Prem Jain, Kangwarn Chinthammit, Satya Akella, Jeff Silberman, David Clear, and Shane Corban have provided many valuable comments.
- Chris Ratcliffe helped me organize my ideas and my terminology. He is a master in messaging.
- Mike Galles, Francis Matus, and Georges Akis kept me honest on all the hardware discussions.
- Krishna Doddapaneni made countless measurements on the performance of different technologies and summarized the results for this book.
- The members of the Pensando Advisory Board for many technically challenging discussions.
- Dinesh Dutt is a great friend who has helped me define the structure of this book.
- Alfredo Cardigliano provided material and insights on DPDK.
- Nital Patwa helped me to better understand the implications of integrating ARM cores in an SoC.
- Elese Orrell and Darci Quack are the graphic designers. They were effortless to work with, and I like the result.
- Brenda Nguyen and Rhonda Biddle for their support.
- Wikipedia made writing this book so much easier that I will donate the first $1,000 of royalties to this fantastic source of information.
- All the other people who gave me advice, suggestions, ideas, and reviews: my deeply felt thank you.

Chapter 1

Introduction to Distributed Platforms

In the last ten years, we have observed an increasingly rapid transition from monolithic servers to virtualization. Initially, this happened inside enterprise networks, creating the need for virtual networking, but it has quickly evolved into modern cloud architectures that add the dimension of multitenancy and, with multitenancy, increased demand for security. Each user requires network services, including firewalls, load balancers, virtual private networks (VPNs), microsegmentation, encryption, and storage, and needs to be protected from other users.

This trend is very evident in cloud providers, but even larger enterprises are structuring their networks as private clouds and need to secure network users from each other.

Software-based services are often the solution. The server CPU implements a Distributed Services Architecture in software. A virtual machine or a container comprises the software that implements the service architecture. All network traffic goes through this software and, after the appropriate processing, packets are delivered to their final destinations (other virtual machines or containers). Similar processing happens on the reverse path.

A pure software solution is limited in performance, and it has high latency and jitter. Moreover, it is very problematic in bare-metal environments where the entire server is dedicated to a user or an application, and there is no place to run the services architecture.

> A distributed services platform is a set of components unified by a management control plane that implements standard network services, such as stateful firewall, load balancing, encryption, and overlay networks, in a distributed, highly scalable way with high performance, low latency, and low jitter. It has no inherent bottleneck and offers high availability. Each component should be able to implement and chain together as many services as possible, avoiding unnecessary forwarding of packets between different boxes that perform different functions. The management control plane provides role-based access to various functions and is itself implemented as a distributed software application.

We offer a new term, *distributed services node (DSN)*, to describe the entity running various network and security services. A DSN can be integrated into existing network components such as NICs (network interface cards), switches, routers, and appliances. The architecture also allows for a software implementation of the DSN, even though only hardware is capable of providing the security and performance needed by today's networks.

Keeping DSNs closer to applications provides better security; however, DSNs should be ideally implemented at a layer that is immune to application, operating system, or hypervisor compromise.

Having multiple DSNs, as distributed as possible, increases scalability dramatically and effectively removes bottlenecks.

This architecture is practical only in the presence of a management system capable of distributing and monitoring service policies to all DSNs.

Figure 1-1 provides a graphical representation of a distributed services platform.

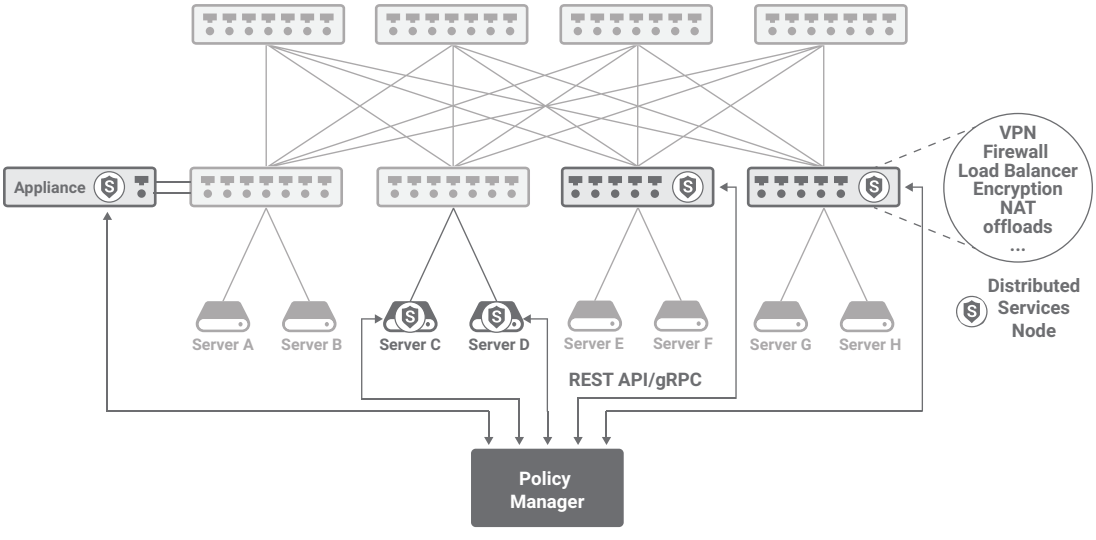


FIGURE 1-1 A Distributed Services Platform

1.1 The Need for a Distributed Services Platform

A real distributed services platform should solve not only performance issues but should also provide:

- A consistent services layer common to bare-metal servers, virtual machines, and containers
- Pervasive security without any entitlements within the perimeter; that is, decouple security from network access

- A security solution that is immune to compromised OSes or hypervisors
- Services orchestration and chaining to simplify management while enabling the delivery of different combinations of services
- Better utilization of resources, higher performance, lower latency, and latency isolation
- Tools capable of troubleshooting the network flows going through multiple services
- Built-in telemetry for edge-to-edge network troubleshooting, rather than debugging individual systems, applications and segments, to give the infrastructure the ability to proactively report potential issues and offending actors
- A comprehensive set of infrastructure services that are easy to manage and that can be used together, including features such as microsegmentation, load balancing, a firewall, encryption service, storage virtualization, and infrastructure services such as RDMA and TCP/TLS proxy
- Programmability in the management, control, and data planes so that software-defined features can be rolled out without requiring hardware swapout or extended hardware development and release cycles

1.2 The Precious CPU Cycles

In recent years, single-thread performance has only grown a few percentage points a year due to the slowdown in Moore's law as well as Dennard scaling issues (see Chapter 7, "CPUs and Domain-Specific Hardware"). Similarly, increasing the number of cores per CPU only partially helps, due to the problems pointed out in Amdahl's law on parallelization.

Another important aspect is that CPU architectures and their associated operating systems are not the best matches for implementing services at the packet level: an example is interrupt-moderation, which reduces the number of interrupts to increase throughput, but has the side effect of jitter explosion.

As processors become more complex, processor cycles are becoming more precious every day and should be used for user applications and not for network services. A purely software-based solution might use a third of the available cores on an enterprise-class CPU to implement services; this is unacceptable in cases of high load on servers. It creates a big pushback for software-based services architectures.

1.3 The Case for Domain-Specific Hardware

Domain-specific hardware can be designed to be the best implementation for specific functions. An example of successful domain-specific hardware is the graphic processor unit (GPU).

GPUs were born to support advanced graphics interfaces by covering a well-defined domain of computing—matrix algebra—which is applicable to other workloads such as artificial intelligence (AI)

and machine learning (ML). The combination of a domain-specific architecture with a domain-specific language (for example, CUDA and its libraries) led to rapid innovation.

Another important measure that is often overlooked is power per packet. Today's cloud services are targeted at 100 Gbps, which for a reasonable packet size is equivalent to 25 Mpps. An acceptable power budget is 25 watts, which equates to 1 microwatt per packet per second. To achieve this minuscule amount of power usage, selecting the most appropriate hardware architecture is essential. For example, Field-Programmable Gate Arrays (FPGAs) have good programmability but cannot meet this stringent power requirement. You might wonder what the big deal is between 25 watts and 100 watts per server. On an average installation of 24 to 40 servers per rack, it means saving 1.8 to 3.0 kilowatts of power per rack. To give you an example, 3 kilowatts is the peak consumption of a single-family home in Europe.

When dealing with features such as encryption (both symmetric and asymmetric) and compression, dedicated hardware structures explicitly designed to solve these issues have much higher throughput and consume far less power than general-purpose processors.

This book should prove to the reader that a properly architected domain-specific hardware platform, programmable through a domain-specific language (DSL), combined with hardware offload for compression and encryption, is the best implementation for a DSN.

Although hardware is an essential aspect of a distributed services platform, a distributed services platform also uses a considerable amount of software.

The management and control planes are entirely software, and even the data plane must be software defined. The hardware is what provides the performance and lowers latency and jitter when used in conjunction with software. When we compare performance and delays, the differences can be enormous. Leading solutions exist in which the first packet of a flow incurs a 20-millisecond delay and other solutions in which the same packet is processed in 2 microseconds: a difference of four orders of magnitude.

1.4 Using Appliances

Today the most common implementation of services is through appliances, typically deployed centrally. These network devices implement services such as firewall, load balancing, and VPN termination. These are discrete boxes, and the traffic is sent to them explicitly in a technique called *tromboning* (see section 2.7). These devices become natural bottlenecks for traffic and impose a weird routing/forwarding topology. They are very high-cost, high-performance devices, and even the most capable ones have limitations in performance when compared to the amount of traffic that even a small private cloud can generate. These limitations, plus the fact that a packet must traverse the network multiple times to go through service chaining, result in reduced throughput and high latency and high jitter.

A distributed services platform avoids these large centralized appliances and relies on small high-performance distributed services nodes (DSNs) located as closely as possible to the final applications they serve. They are also multifunctional; that is, they implement multiple services and can chain them internally, in any order, without needing to traverse the network numerous times.

1.5 Attempts at Defining a Distributed Services Platform

The first question we should ask ourselves is, "Which services should this architecture support?" A precise classification is difficult, if not impossible, but Figure 1-2 is an example of some of the services typically associated with a domain-specific platform.

From 10,000 feet, we can see two groups of services: infrastructure services and value-added services.

Infrastructure services are things such as Ethernet and bridging, IP and routing, storage access through a modern protocol like NVMe, RDMA transport, TCP termination, and overlay network processing.

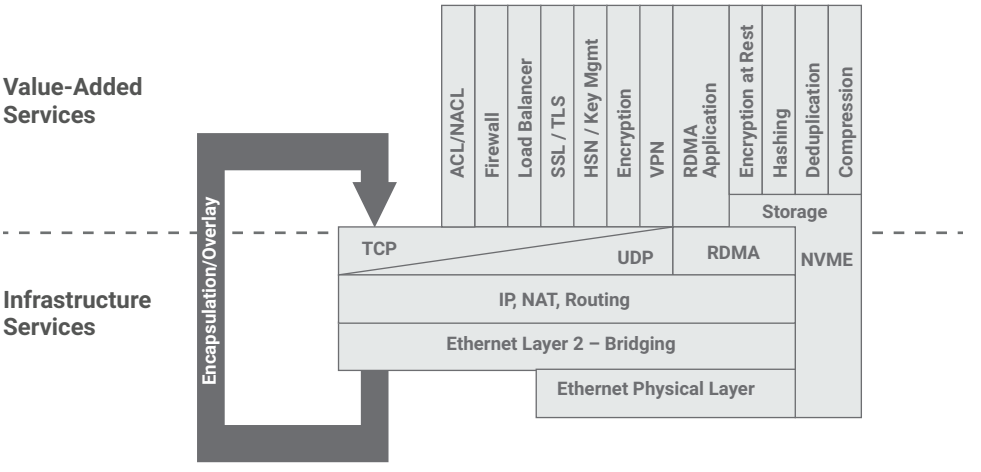


FIGURE 1-2 Services

Value-added services include a firewall, load balancer, encryption (both symmetric and asymmetric, both in flight and at rest), key management and secure storage, classical VPNs like IPsec, and more modern ones such as SSL/TLS, storage compression, and deduplication.

In this book, we will present infrastructure and value-added services together because they are deployed together in the majority of the cases.

The first attempt at a truly "distributed" network architecture can be traced back to software-defined networking (SDN). SDN is a paradigm that was introduced initially on switches and routers but was expanded to servers. It offers the capability to control and program the NIC virtual forwarding functions.

The main focus of SDNs is on infrastructure services. They do not currently address value-added services, but they create a framework where DSNs are under the central coordination of a common manager and work together toward a common goal.

There is also an open-source effort within the container community, called *service mesh*, that defines the services, such as load balancing, telemetry and security, distributed across the clusters of nodes

and combined with a management control plane. The approach uses a software proxy sitting next to an application to provide layer 4 or layer 7 load balancing features and TLS security between applications and provide telemetry for all the traffic that passes through the proxy. The management control plane provides integration with orchestration systems, such as Kubernetes, and also provides a security framework to do key management for applications and define authorization primitives that can police interapplication communication. Although the effort started for containers, the concepts and code can be leveraged for virtual machines. There are many implementations of service mesh, such as Istio, Nginx, Linkerd, and some commercial closed-source implementations.

It is definitely possible to provide a much superior service mesh with DSNs by offering better security via keeping private keys within the hardware root of trust, by improving performance by an order of magnitude, and by reducing interapplication latency, without losing the software programmability.

The distributed services platform also attempts to address a few additional elements:

- Provide immunity from host/application/hypervisor compromises; that is, the enforcer shouldn't be compromised if the enforcee is compromised
- Provide services beyond the network, for example, storage, RDMA, and so on
- Offer low latency, high throughput, and latency isolation without impacting application performance
- Exhibit cloudlike scale to handle millions of sessions

A distributed services platform may be used alongside service mesh, which is an application layer concept, as opposed to an infrastructure layer concept. For example, the distributed services platform may provide isolation, security, and telemetry at the virtualization infrastructure layer, whereas service mesh can provide application layer TLS, API routing, and so on.

1.6 Requirements for a Distributed Services Platform

A truly distributed services platform requires the availability of DSNs that are placed as closely as possible to the applications. These DSNs are the enforcement or action points and can have various embodiments; for example, they can be integrated into NICs, appliances, or switches. Having as many services nodes as possible is the key to scaling, high performance, and low delay and jitter. The closer a DSN is to applications, the lesser the amount of traffic it needs to process, and the better the power profile becomes.

Services may appear to be well defined and not changing over time, but this is not the case. For example, new encapsulations or variations of old ones or different combinations of protocols and encapsulations are introduced over time. For this reason, DSNs need to be programmable in the management, control, and data planes. The control and management planes may be complicated, but they are not data intensive and are coded as software programs on standard CPUs. Data plane programmability is a

crucial requirement because it determines the performance and the scaling of the architecture. Network devices that are data plane programmable are still pretty rare, even if there have been some attempts in the adapter space with devices that are typically called SmartNIC and in the switching/routing space using a domain-specific programming language called P4.

An excellent services platform is only as good as the monitoring and troubleshooting features that it implements. Monitoring has significantly evolved over the years, and its modern version is called *telemetry*. It is not just a name change; it is an architectural revamp on how performance is measured, collected, stored, and postprocessed. The more dynamic telemetry is and the less likely it is to introduce latency, the more useful it is. An ideal distributed services platform has "always-on telemetry" with no performance cost. Also, compliance considerations are becoming extremely important, and being able to observe, track, and correlate events is crucial.

Where do services apply? To answer this question, we need to introduce a minimum of terminology. A common way to draw a network diagram is with the network equipment on top, and the compute nodes at the bottom. If you superimpose a compass rose with the North on top, then the term North-South traffic means traffic between the public network (typically the Internet) and servers; the term East-West implies traffic between servers (see Figure 1-3).

Historically the North-South direction has been the focus of services such as firewall, SSL/TLS termination, VPN termination, and load balancing. Protecting the North-South direction is synonymous with protecting the periphery of the cloud or data center. For many years, it has been security managers' primary goal because all the attacks originated on the outside, and the inside was composed of homogeneous, trusted users.

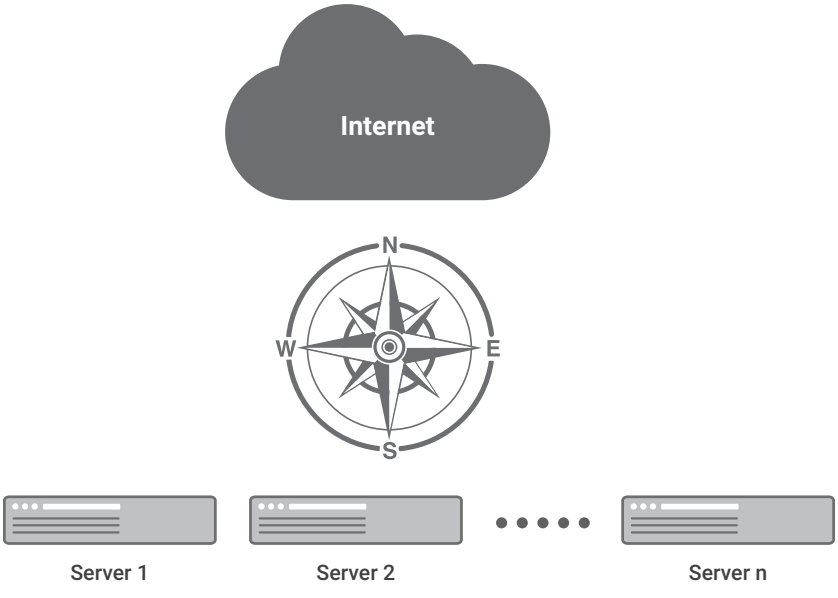


FIGURE 1-3 North-South vs. East-West

With the advent of public clouds, the change in the type of attacks, the need to compartmentalize large corporations for compliance reasons, the introduction of highly distributed microservice architectures, and remote storage, East-West traffic is now demanding the same level of services as the North-South connections.

East-West traffic requires better services performance than North-South for the following reasons:

- Usually, the North-South traffic has a geographical dimension; for example, going through the Internet creates a lower bound to the delay of milliseconds, due to propagation delays. This is not the case for East-West traffic.
- East-West traffic is easily one order of magnitude higher in bytes than North-South traffic, a phenomenon called "traffic amplification," where the size of the response and internal traffic can be much larger, that is, "amplified," compared to the inbound request. For this reason, it requires services with higher throughput.
- With the advent of solid-state disks (SSDs), the storage access time has dramatically decreased, and delays associated with processing storage packets must be minimal.
- In microservice architectures, what on the North-South direction may be a simple transaction is in reality composed of multiple interactions between microservices on the East-West direction. Any delay is critical because it is cumulative and can quickly result in performance degradation.

Institutions with sensitive data, such as banks or healthcare providers, are considering encrypting all the East-West traffic. It implies, for instance, that each communication between two microservices must be encrypted and decrypted: If the encryption service is not line-rate and low-latency, this will show up as degraded performance.

1.7 Summary

This introductory chapter delineated what a distributed services platform could be, the continuously evolving requirements of the cloud world, the rising importance of the East-West traffic, and the need for domain-specific hardware and common management.

Starting with the next chapter, we will cover all these aspects in detail.

Index

Numbers

3D Xpoint, 120
7 nm circuit-manufacturing process, 178
16 nm circuit-manufacturing process, 177–178
802.1Q tag, 12

A

Access-Aggregation-Core model, 10
ACLs (access control lists), 28, 157
active-active mode, 39
active-standby mode, 39
adapters, DSN, 164–166
adaptive logic modules (ALMs), 182
ADD API (CNI), 50
AES (Advanced Encryption Standard), 89, 125
AI (artificial intelligence), 4–5
ALGs (application layer gateways), 79
Alibaba Cloud, 36, 46
ALMs (adaptive logic modules), 182
alpha-quality reference compiler (P4), 202
Amazon Web Services (AWS), 36, 46
AMD
Epyc processor, 137
xenproject.org, 46
Amdahl, Gene, 135
Amdahl's Law, 135–136
Ansible, 57, 212
APIs (application programming interfaces), 215–217
Apple A12 Bionic, 178

appliances, 5
distributed services platforms in, 171–172
tromboning, 5, 157
example of, 25
hybrid, 27
with VRF (virtual routing and forwarding), 26–27
with VXLAN (Virtual Extensible LAN), 25–26
application layer gateways (ALGs), 79
application programming interfaces (APIs), 215–217
application-specific integrated circuits (ASICs), 183–184
ARM
CPU cores, 179–181
xenproject.org, 46
***The Art of Invisibility* (Mitnick), 95**
artificial intelligence (AI), 4–5
ASICs (application-specific integrated circuits), 183–184
asymmetric encryption, 89–90
atomic operations (RDMA), 108–109
auditing (API), 217
AWS (Amazon Web Services), 36, 46
Azure, 36
GFT (generic forwarding table), 29–30
SmartNIC, 154–155

B

back end of line (BEOL), 178
Baldi, Mario, 202
bandwidth requirements, 186–187
Barefoot Networks Tofino switch, 202
bare-metal hypervisors, 38–39
Hyper-V, 41–43
VMware ESXi, 40–41
XEN, 46
bare-metal servers, 161
Batten, C., 131
BEOL (back end of line), 178
Berkeley Packet Filter (BPF), 76
BitDefender, 46
Blade servers, 146
bonding, 40
boot systems, secure, 97
Borg, 50
BPF (Berkeley Packet Filter), 76
bridging and routing. *See* routing
Broadcom BCM58800, 142
brownfield projects, 162
bump-in-the-wire, 168–169
buses, 143–144

C

C++, 208
cache
cache-based forwarding, 27–29
DSN requirements, 175
flow, 28
CAD (computer-aided design), 183
CAs (certificate authorities), 210
Cassandra, 212–213
CEE (Converged Enhanced Ethernet), 151
central processing units. *See* CPUs (central processing units)
centralized routing, 66
certificate authorities (CAs), 210
certificates, digital, 90
ChaCha20-Poly1305, 89
CHECK API (CNI), 50
checksum offload, 150
Chef, 212
Cinder module (OpenStack), 56
Citrix, 46
Clos, Charles, 14
Clos topology, 14–15
Cloud Native Computing Foundation. *See* CNCF (Cloud Native Computing Foundation)
cloud providers, 164
clouds. *See also individual cloud services*
hybrid, 36–37
private, 36
public, 36
virtualization and, 35–37
CMDB (configuration management database), 218

CNCF (Cloud Native Computing Foundation), 207–208
containerd, 48
Kubernetes, 50–52
CNI (Container Network Interface), 49–50
CNP (congestion notification packet), 114
CockroachDB, 212–213
commands, ping, 181
communication interfaces, 100–102
Compaq, 103
computer-aided design (CAD), 183
configuration, declarative, 206–207
configuration management database (CMDB), 218
congestion management signaling, 109
congestion notification packet (CNP), 114
constraints, distributed services platforms, 161–163
drivers, 162
greenfield versus brownfield deployment, 162
PCIe-only services, 162–163
power budget, 163
virtualized versus bare-metal servers, 161
Container Network Interface (CNI), 49–50
Container Storage Interface (CSI), 50
containerd, 48
containers
CNI (Container Network Interface), 49–50
container runtimes, 48
CSI (Container Storage Interface), 50
Kata Containers, 49
Kubernetes, 50–52
LXC (LinuX Containers), 47
overview of, 47–48
control plane, distributed management. *See* distributed management control plane
controls (P4), 193
Converged Enhanced Ethernet (CEE), 151
converged fabric, support for, 151
Core OS, Rocket, 48
COT (customer-owned tooling), 177
CPUs (central processing units)
Amdahl's Law, 4
demands on, 4
Dennard scaling, 4, 134–135
Distributed Services Architecture implementation, 2
DSN requirements, 176
general-purpose, 131
historical perspective of, 130–131
historical trend data, 131–132
loads, 159
Moore's law, 4, 103, 132–134
sea of cores solution, 179–181
single-thread performance, 137–138
technical factors limiting, 136–137
Create, Read, Update, and Delete (CRUD) operations, 213
CRUD (Create, Read, Update, and Delete) operations, 213
Cryptographic Protection on Block Storage Devices, 125
CSI (Container Storage Interface), 50
customer-owned tooling (COT), 177

D

DAG (directed acyclic graph), 193
Data Center Bridging (DCB), 112, 151
Data Center Bridging eXchange (DCBX), 112, 151
Data Center Ethernet (DCE), 151
Data Center Quantized Congestion Notification (DCQCN), 114
data field (Ethernet packets), 12
Data Plane Development Kit (DPDK), 68
data plane model of storage protocols, 120–122
data replication, 214
data types (P4), 193
Datagram TLS (DTLS), securing tunnels with, 94
DCB (Data Center Bridging), 112, 151
DCBX (Data Center Bridging eXchange), 112, 151
DCE (Data Center Ethernet), 151
DCQCN (Data Center Quantized Congestion Notification), 114
DDR4, 186
DDR5, 186

declarative configuration model, 206–207
dedicated networks, hosts connected to, 100–101
Deficit Weighted Round Robin (DWRR), 150
DEL API (CNI), 50
Dennard, Robert, 134
Dennard scaling, 4, 134–135
deparsers (P4), 191
deployment
distributed management control plane, 211–212
RDMA (Remote Direct Memory Access), 110–112
destination MAC address field (Ethernet packets), 12
Diffie, Whitfield, 89
digital certificates, 90
Digital Equipment Corporation, 103
direct memory access (DMA), 101
directed acyclic graph (DAG), 193
distributed firewalls, 85–86
distributed management control plane, 204
API architecture, 215–217
architectural traits of, 205–206
building as cloud-native application, 207–208
control plane acceleration, 117–118
declarative configuration, 206–207
deployment, 211–212
failure handling, 214–215
federation
concept of, 220–223
distributed multiple SDSPs, 220
overview of, 218
single SDSP scaling, 219
monitoring and troubleshooting, 209
performance of, 212–214, 223–226
scale of, 212–214, 223–226
securing, 210–211
distributed multiple SDSPs (software defined services platforms), 220
Distributed Services Architecture, 2
distributed services nodes. *See* DSNs (distributed services nodes)
distributed services platforms
appliances, avoidance of, 5
architecture of, 2–3
constraints, 161–163
drivers, 162
greenfield versus brownfield deployment, 162
PCIe-only services, 162–163
power budget, 163
virtualized versus bare-metal servers, 161
CPU architecture and, 4
distributed management control plane
API architecture, 215–217
architectural traits of, 205–206
building as cloud-native application, 207–208
declarative configuration, 206–207
deployment, 211–212
failure handling, 214–215
federation, 218–223
monitoring and troubleshooting, 209
overview of, 204
performance of, 212–214, 223–226
scale of, 212–214, 223–226
securing, 210–211
domain-specific hardware
concept of, 139
historical perspective of, 130–131
need for, 4–5, 139
DSNs (distributed services nodes). *See also* CPUs (central processing units)
advantages of, 5
building blocks of, 174–176
defined, 3
hardware architectures, 174–187
implementation of, 130
goals for, 156–161
host mode versus network mode, 160–161
low jitter, 158–159
low latency, 158
manageability, 160
minimal CPU load, 159
observability and troubleshooting capacity, 159–160
PCIe firewall, 161
scaling, 157
services everywhere, 157
speed, 158

implementation of
in appliances, 171–172
bump-in-the-wire, 168–169
in domain-specific hardware inside NIC, 166–167
in software, 164–166
summary of, 172
in switch, 169–170
need for, 3–4
overview of, 2–3, 156
requirements for, 7–9
services
overview of, 6
service mesh, 6–7
target users, determining, 163–164

distributed storage services, 126–127
DMA (direct memory access), 101
DNS (Domain Name Server), 29
Docker, 48, 208
Domain Name Server (DNS), 29
domain-specific hardware
concept of, 139
historical perspective of, 130–131
need for, 4–5, 139
domain-specific hardware inside NIC, 164–166
domain-specific language (DSL), 5, 193
DPDK (Data Plane Development Kit), 68
DPDK Generic Flow API, 74
DRAM (Dynamic RAM)
on-chip DRAM, 186
external DRAM, 186
memory needs, 185
drivers, 162
DS platforms. *See* distributed services platforms
DSL (domain-specific language), 5, 193
DSNs (distributed services nodes). *See also* CPUs (central processing units)
advantages of, 5
building blocks of, 174–176
defined, 3
hardware architectures
ASICs (application-specific integrated circuits), 183–184
choosing, 178–179
DSN building blocks, 174–176
DSN power consumption, 184–185
FPGAs (field-programmable gate arrays), 181–183
memory, 185–187
overview of, 174
sea of cores solution, 179–181
silicon design, 176–178
implementation of, 130
DTLS (Datagram TLS), securing tunnels with, 94
Dutt, Dinesh, 15
DWRR (Deficit Weighted Round Robin), 150
dynamic bonding, 40
Dynamic RAM. *See* DRAM (Dynamic RAM)

E

East-West traffic, 8–9, 84
eBPF (extended Berkeley Packet Filter), 76
ECMP (equal cost multi-path), 14, 20
ElasticDB, 212–213
ELK, 212
elliptic curves, 89–90
encapsulation
generic, 17–18
GRE (Generic Routing Encapsulation), 18–19
IP in IP, 18
modern, 19
modern encapsulations, 19
MTU (maximum transmission unit) considerations, 22
termination of, 23
tunnel security, 22–23
VXLAN (Virtual Extensible LAN), 19–22
encryption, 22–23
asymmetric, 89–90
Cryptographic Protection on Block Storage Devices, 125
digital certificates, 90
hashing, 90
PUF (physical unclonable function), 91
secure key storage, 90–91

secure tunnels, 92
with DTLS (Datagram TLS), 94
with IPsec, 92–93
with TLS (Transport Layer Security), 93–94
storage security with, 125
symmetric, 89
TCP/TLS/HTTP implementation, 91
endpoints, tunnel, 17
Enhanced Transmission Selection (ETS), 151
Enlightened I/O, 42
enterprise data centers, 163–164
Envoy, 208
equal cost multi-path (ECMP), 14, 20
Esmaeilzadeh, Hadi, 135
ESXi (VMware), 40–41
Etcd, 212–213
Ethernet packets, 12. *See also* routing
Ethernet ports, DSN requirements for, 174
Ethernet VPN (EVPN), 22, 95
Ethertype field (Ethernet packets), 12
ETS (Enhanced Transmission Selection), 151
ETSI (European Telecommunications Standards Institute), 57
EVPN (Ethernet VPN), 22, 95
eXpress Data Path (XDP), 76–77
expressions (P4), 193
extended Berkeley Packet Filter (eBPF), 76
extending P4 (Programming Protocol-independent Packet Processors), 201–202
language composability, 201–202
PNA (Portable NIC Architecture), 201
extern objects (P4), 194
external DRAM, 186

F

Fabrics, NVME over, 120
failure handling, 214–215
Fairchild Semiconductor, 132
Fast Data - Input/Output (FD.io), 75
FCS (frame check sequence), 12
FD.io (Fast Data - Input/Output), 75
Federated Service Manager (FSM), 220–221
federation
concept of, 218, 220–223
distributed multiple SDSPs, 220
single SDSP scaling, 219
FIB (forwarding information base), 13–14, 68
Fibre Channel, 119
Fielding, Roy, 54
field-programmable gate arrays (FPGAs), 5, 30, 142, 181–183
Finagle, 54, 208
firewalls
distributed, 85–86
PCI Express, 161
flow cache, 28
flow dissector, 73
form factors, NIC (network interface card)
LOM (LAN On Motherboard), 148–149
OCP (Open Compute Project) mezzanine cards, 147
PCI plug-in cards, 144–145
proprietary mezzanine cards, 146
forwarding
cache-based, 27–29
FIB (forwarding information base), 13–14, 68
GFT (generic forwarding table), 29–30
hash-based, 39
L2, 12
L3, 12–13
LPM (longest prefix match) forwarding, 13–14
vEth-based, 39
VRF (virtual routing and forwarding), 14
FPGAs (field-programmable gate arrays), 5, 30, 142, 181–183
frame check sequence (FCS), 12
frames, 12. *See also* routing
FSM (Federated Service Manager), 220–221
full virtualization, 39
functions
physical, 152, 175
virtual, 152, 175
FutureIO, 103

G

GCM (Galois Counter Mode), 89
GDDR6, 186
general-purpose CPUs (central processing units), 131
generic forwarding table (GFT), 29–30
Generic Routing Encapsulation (GRE), 18–19
generic segmentation offload (GSO), 150
GFT (generic forwarding table), 29–30
GigaBytes per second (GB/s), 185
Glance module (OpenStack), 56
"Go Back N" approach, 115–116
Golang, 208
Google
Cloud Platform, 36
OpenFlow at, 66
GPUs (graphic processor units), 4
Grafana, 212
graphic processor units (GPUs), 4
gRBC Routing Information Base Interface (gRIBI), 67–68
GRE (Generic Routing Encapsulation), 18–19
greenfield projects, 162
gRIBI (gRBC Routing Information Base Interface), 67–68
gRPC, 54–55, 160, 208, 210
gRPC-lb, 54
GSO (generic segmentation offload), 150

H

Hammond, L., 131
hardware architectures (DSN)
ASICs (application-specific integrated circuits), 183–184
choosing, 178–179
DSN building blocks, 174–176
DSN power consumption, 184–185
FPGAs (field-programmable gate arrays), 181–183
memory
bandwidth requirements, 186–187
on-chip DRAM, 186
external DRAM, 186
host memory, 185
memory needs, 185
overview of, 174
sea of cores solution, 179–181
silicon design
7 nm process, 178
16 nm process, 177–178
"silicon sweet-spot," 176–177
hardware description language (HDL), 182
hardware security modules (HSMs), 90–91, 157
hash-based forwarding, 39
Hash-based Message Authentication (HMAC), 96
hashing, 90, 126
HBM (high-bandwidth memory), 186
HCI (hyper-converged infrastructure), 119
HDL (hardware description language), 182
Hellman, Martin, 89
Helm, 211–212
high-bandwidth memory (HBM), 186
high-performance computing (HPC), 110–111
HMAC (Hash-based Message Authentication Code), 96
Horowitz, M., 131
host memory, 185
host mode, 160–161
hosted hypervisors, 38–39
hosts connected to dedicated networks, 100–101
hosts with unified networks, 100–102
host-to-site VPNs (virtual private networks), 96
HP, 103
HPC (high-performance computing), 110–111
HSMs (hardware security modules), 90–91, 157
HTTP (Hypertext Transfer Protocol), 88, 91
HTTPS (HTTP Secure), 88
Huawei, xenproject.org, 46
Huawei Kirin 980, 178
Huffman Coding, 125
hybrid clouds, 36–37
hybrid tromboning, 27

hyper-converged infrastructure (HCI), 119
Hypertext Transfer Protocol (HTTP), 88, 91
Hypertext Transfer Protocol Secure (HTTPS), 88
Hyper-V, 29–30, 41–43
hypervisors
Hyper-V, 41–43
KVM (Kernel-based Virtual Machine), 36, 43–46
overview of, 37–40
QEMU (Quick EMUlator), 43
types of, 38–39
VMware ESXi, 40–41
XEN, 46

I

i7–5557U Intel processor, 177
IBM, 36, 103
IBTA (InfiniBand Trade Association), 103
IC (integrated circuit) manufacturing processes
7 nm process, 178
16 nm process, 177–178
"silicon sweet-spot," 176–177
IDL (interface description language), 55
IEEE (Institute of Electrical and Electronics Engineers), 11
Cryptographic Protection on Block Storage Devices standard, 125
IEEE 802.1, 11
IETF (Internet Engineering Task Force), 11
implementation of distributed services platforms
in appliances, 171–172
bump-in-the-wire, 168–169
in domain-specific hardware inside NIC, 166–167
in software, 164–166
summary of, 172
in switch, 169–170
In-band Network Telemetry (INT), 193
InfiniBand RDMA. *See* RDMA (Remote Direct Memory Access)
InfiniBand Trade Association (IBTA), 103
InfluxDB, 212–213
in-service upgrades, 215
Institute of Electrical and Electronics Engineers. *See* IEEE (Institute of Electrical and Electronics Engineers)
INT (In-band Network Telemetry), 193
INT header (P4), 201
integrated circuit. *See* IC (integrated circuit) manufacturing processes
Intel
Agilex AGF 008, 182
x86 processors, 130
xenproject.org, 46
***Interconnections: Bridges and Routers* (Perlman), 63**
interface description language (IDL), 55
Intermediate-System to Intermediate-System (IS-IS), 15
internal layer 2 switches, 175
International System of Units, 184
International Technology Roadmap for Semiconductors (ITRS), 135
Internet Engineering Task Force (IETF), 11
Internet of Things (IoT), 57, 87
Internet Protocol. *See* IP (Internet Protocol)
interrupt moderation, 151
I/O offloads, 130
IoT (Internet of Things), 57, 87
IP (Internet Protocol)
IP in IP encapsulation, 18
IP masquerading, 79
IPsec, 92–93
Ironic module (OpenStack), 56
iSER (iSCSI Extensions for RDMA), 121
IS-IS (Intermediate-System to Intermediate-System), 15
Istio, 6–7
ITRS (International Technology Roadmap for Semiconductors), 135
iWARP, 109–110

J

Jacobson, Van, 76
jitter, 158–159, 181
JWT (JSON Web Tokens), 216–217

K

Kafka, 212–213
Kata Containers, 49
kernal bypass, 104
Kernel-based Virtual Machine (KVM), 36, 43–46
keys, secure key storage, 90–91
Keystone module (OpenStack), 56
Kibana, 212
kubelet, 50
kube-proxy, 50
Kubernetes, 50–52, 208, 211–212
KVM (Kernel-based Virtual Machine), 36, 43–46

L

L2 forwarding, 12
L3 forwarding, 12–13
Labonte, F., 131
LAN On Motherboard (LAN), 148–149
language composability, 201–202
large receive offload (LRO), 150
large send offload (LSO), 150
latency, 158, 181
LDAP (Lightweight Directory Access Protocol), 210–211
Lempel-Ziv (LZ) compression, 125
LEs (logical elements), 182
libcontainer, 48
libvirt, 45
Lightweight Directory Access Protocol (LDAP), 210–211
Linkerd, 6–7
LinuX Containers (LXC), 47
load balancing, 79–80, 208
loads (CPU), 159
logical elements (LEs), 182
LOM (LAN On Motherboard), 148–149
longest prefix match (LPM) forwarding, 13–14
look up table (LUT), 182
lossy networks, RoCEv2 and, 112–117
LPM (longest prefix match) forwarding, 13–14
LRO (large receive offload), 150
LSO (large send offload), 150
LUT (look up table), 182
LXC (LinuX Containers), 47
LZ (Lempel-Ziv) compression, 125
LZ4, 125

M

machine learning (ML), 4–5
manageability, distributed services platforms, 160
management architecture. *See* distributed management control plane
masquerading (IP), 79
match-action pipeline, 191
maximum transmission unit (MTU), 22
McCanne, Steven, 76
McKeown, Nick, 63, 66, 190
MD5 algorithm, 90
Mega Transactions per second (MT/s), 185
Mellanox Technologies, 112
Meltdown, 84, 90–91
memory, 127
bandwidth requirements, 186–187
on-chip DRAM, 186
external DRAM, 186
HBM (high-bandwidth memory), 186
host, 185
memory needs, 185
RDMA (Remote Direct Memory Access), 175
advantages of, 104–106
architecture of, 106–107
control plane acceleration, 117–118
data security, 118–119
deployments, 110–112
history of, 103
operations, 108–109
remote nonvolatile memory access, 118
RoCE (RDMA over Converged Ethernet), 19, 109–110, 112–117
scalability, 109
transport services, 108
registration, 107
TCAM (ternary content-addressable memory), 139
Memory Management Units (MMUs), 161

Mesos, 48, 208
Message Signaled Interrupts-Extended (MSI-X), 151
metal-oxide-semiconductor field-effect transistors (MOSFETs), 136–137. *See also* transistor count
mezzanine cards
OCP (Open Compute Project) mezzanine, 147
proprietary, 146
microprocessors. *See* CPUs (central processing units)
microsegmentation, 86–87
microservice architecture
gRPC, 54–55
overview of, 52–54
REST API, 54
Microsoft Azure, 36
GFT (generic forwarding table), 29–30
SmartNIC, 154–155
Microsoft Hyper-V, 29–30, 41–43
Miller, David, 76
Minio, 212–213
MIPS cores, 179
ML (machine learning), 4–5
MMUs (Memory Management Units), 161
modern encapsulations, 19
MongoDB, 212–213
monitoring distributed management control plane, 209
Moore, Gordon, 132, 138
Moore's law, 4, 103, 132–134, 138
MOSFETs (metal-oxide-semiconductor field-effect transistors), 136–137. *See also* transistor count
MPLS (Multiprotocol Label Switching), 23–24, 95
MSI-X (Message Signaled Interrupts-Extended), 151
MTU (maximum transmission unit), 22
multicast handling, 150
multicore CPU architecture, 130
Multiprotocol Label Switching (MPLS), 23–24, 95
multitenancy, 37

N

NAPI (New API), 150
NAT (network address translation), 157
ALGs (application layer gateways), 79
stateful, 79
National Institute of Standards and Technology (NIST), 89
NATS, 212–213
ndo-setup-tc utility, 73
NetFlow, 81
network address translation (NAT), 79, 157
network design. *See also* NICs (network interface cards)
Access-Aggregation-Core model, 10
bridging and routing
defined, 11
L2 forwarding, 12
L3 forwarding, 12–13
LPM (longest prefix match) forwarding, 13–14
VRF (virtual routing and forwarding), 14
cache-based forwarding, 27–29
Clos topology, 14–15
East-West traffic, 8–9, 84
encapsulation termination, 23
GFT (generic forwarding table), 29–30
lossy networks, RoCEv2 and, 112–117
North-South traffic, 8–9, 84
overlays
architecture of, 16–18
generic encapsulation, 17–18
GRE (Generic Routing Encapsulation), 18–19
IP in IP encapsulation, 18
modern encapsulations, 19
MTU (maximum transmission unit) considerations, 22
support for, 151
tunnel endpoints, 17
tunnels and tunnel endpoints, 17
VXLAN (Virtual Extensible LAN) encapsulation, 19–22
overview of, 10–11
redundancy, 214

secure tunnels, 22–23, 92
with DTLS (Datagram TLS), 94
with IPsec, 92–93
with TLS (Transport Layer Security), 93–94
SR (segment routing), 23–24
telemetry, 80–81
tromboning, 5, 157
example of, 25
hybrid, 27
with VRF (virtual routing and forwarding), 26–27
with VXLAN (Virtual Extensible LAN), 25–26
underlays, 16
Network Function Virtualization (NFV), 57
network interface cards. *See* NICs (network interface cards)
Network Load Balancer (NLB), 208
network mode, 160–161
network on chip (NoC), 175
Network Virtualization using Generic Routing Encapsulation (NVGRE), 18–19
networking services
challenges of, 62–63
load balancing, 79–80
SDN (software-defined networking)
DPDK (Data Plane Development Kit), 68
gRIBI (gRBC Routing Information Base Interface), 67–68
OpenFlow, 64–66
overview of, 63–64
SW-WAN (software-defined wide-area network), 66–67
stateful NAT (network address translation), 79
troubleshooting, 80–81
VPNs (virtual private networks), 94–96
vSwitches
BPF (Berkeley Packet Filter), 76
classification of, 69
DPDK Generic Flow API, 74
eBPF (extended Berkeley Packet Filter), 76
OVS (Open vSwitch), 70–73
summary of, 78–79
tc-flower, 73–74
VPP (Vector Packet Processing), 75
XDP (eXpress Data Path), 76–77
Neutron module (OpenStack), 56
New API (NAPI), 150
NFV (Network Function Virtualization), 57
Nginx, 6–7, 208
NGIO, 103
Nicira, 70
NICs (network interface cards), 100
DSN adapters, 164–166
evolution of, 142, 149–153
form factors
LOM (LAN On Motherboard), 148–149
OCP (Open Compute Project) mezzanine cards, 147
PCI plug-in cards, 144–145
proprietary mezzanine cards, 146
server buses, 143–144
SmartNIC, 154–155
SR-IOV (Single Root Input/Output Virtualization), 149–153
VirtIO (Virtual I/O), 153–154
NIST (National Institute of Standards and Technology), 89
NLB (Network Load Balancer), 208
NOC (network on chip), 175
non-recurring engineering (NRE), 183
Non-Volatile Memory express (NVMe), 119
North-South traffic, 8–9, 84
Nova module (OpenStack), 56
NRE (non-recurring engineering), 183
NSX (VMware), 41, 164
NVDIMMs, 120
NVGRE (Network Virtualization using Generic Routing Encapsulation), 18–19
NVIDIA CP100 GPU, 137
NVMe-oF (NVME over Fabrics), 120
NVMe, 6
NVRAM interface, 176

O

OAS (OpenAPI Specification), 54
observability, distributed services platforms, 159–160
OCI (Open Container Initiative), 48

OCP (Open Compute Project), 147, 163
offloading, 159
stateless offloads, 150
storage services, 126–127
ole-Based Access Control (RBAC), 210–211
Olukotun, K., 131
on-chip DRAM, 186
one-sided operations, RDMA (Remote Direct Memory Access), 104, 106
ONF (Open Networking Foundation), 64–66, 190
Open Compute Project (OCP), 147, 163
Open Container Initiative (OCI), 48
Open Networking Foundation (ONF), 64–66, 190
Open Shortest Path First (OSPF), 15
Open Source NFV Management and Orchestration (MANO), 57
Open vSwitch (OVS), 70–73
OpenAPI Specification (OAS), 54, 160
OpenFlow, 64–66
open-source software
gRPC, 54–55
KVM (Kernel-based Virtual Machine), 43–46
Open Source NFV Management and Orchestration (MANO), 57
OpenStack, 55–57
OpenVPN, 96
OVS (Open vSwitch), 70–73
QEMU (Quick EMUlator), 43
openssl utility, 89
OpenStack, 55–57
***OpenStack Cloud Computing*, 57**
OpenStack Foundation, 55
OpenTracing, 208
OpenVPN, 96
OpenZipkin, 208
operations, RDMA (Remote Direct Memory Access), 108–109
Oracle
Cloud Infrastructure, 36
xenproject.org, 46
orchestration, 48
OSPF (Open Shortest Path First), 15
overlays, network
architecture of, 16–18
generic encapsulation, 17–18
GRE (Generic Routing Encapsulation), 18–19
IP in IP encapsulation, 18
modern encapsulations, 19
MTU (maximum transmission unit) considerations, 22
support for, 151
tunnels and tunnel endpoints, 17
VXLAN (Virtual Extensible LAN) encapsulation, 19–22
OVH, 36
OVS (Open vSwitch), 70–73

P

P4 (Programming Protocol-independent Packet Processors)
example of, 195–199
extending, 201–202
language composability, 201–202
PNA (Portable NIC Architecture), 201
INT (In-band Network Telemetry), 193
overview of, 190–192
P4 INT, 201
P4 language, 193–194
P4 version 16, 192–193
P4Runtime API, 193, 199–200
programming and development tools, 202
PSA (Portable Switch Architecture), 193, 194–195
P4 Language consortium, 194
P4 Language Design Working Group, 202
packet processing graphs (VPP), 75
packets. *See also* routing
filtering, 151
structure of, 12
paravirtualization, 39
parsers (P4), 191, 193
PAT (port address translation), 79
path MTU discovery (PMTUD), 22

PCI (Peripheral Component Interconnect)
DSN requirements, 175
PCI plug-in cards, 144–145
PCIe (PCI Express), 143–144, 161
PCIe-only services, 162–163
Pentium 4 processors, 130
Pentium processors, 130
Pentium Pro/II processors, 130
Per Priority Pause (PPP), 151
perfect forward secrecy (PFS), 88
performance, distributed management control plane, 212–214, 223–226
Peripheral Component Interconnect. *See* PCI (Peripheral Component Interconnect)
Perlman, Radia, 63
Per-Priority Flow Control (PFC), 112
PFC (Per-Priority Flow Control), 112, 151
PFS (perfect forward secrecy), 88
PFs (physical functions), 152
phishing, 84
physical functions (PFs), 152, 175
physical unclonable function (PUF), 91
ping command, 181
PMTUD (path MTU discovery), 22
PNA (Portable NIC Architecture), 201
pods (Kubernetes), 52
port address translation (PAT), 79
Portable NIC Architecture (PNA), 201
Portable Switch Architecture (PSA), 193, 194–195
ports
DSN requirements, 174
OpenFlow, 65
power consumption
constraints on, 163
DSNs (distributed services nodes), 184–185
power budget, 163
PPP (Per Priority Pause), 151
Precision Time Protocol (PTP), 152, 160
Priority-based Flow Control (PFC), 151
private clouds, 36
processors. *See* CPUs (central processing units)
programmability, 182
programmable deparsers, 191
programmable match-action pipeline, 191
programmable parsers, 191
Programming Protocol-independent Packet Processors. *See* P4 (Programming Protocol-independent Packet Processors)
Prometheus, 212–213
proprietary mezzanine cards, 146
protobuf, 54, 160, 200, 208
Protocol 41 encapsulation, 18
protocol buffers, 54. *See also* protobuf
protocol overload, 104, 105
PSA (Portable Switch Architecture), 193, 194–195
PTP (Precision Time Protocol), 152, 160
public clouds, 36
PUF (physical unclonable function), 91
Puppet, 212
Python, 208

Q

QEMU (Quick EMUlator), 43
QoS (Quality of Service)
NICs (network interface cards), 150
QoS marking, 109
Queue Pairs (QPs), 106–107

R

RAID (Redundant Array of Independent Disks), 126
RAW (Raw Architecture Workstation), 130
RBAC (Role-Based Access Control), 210–211
RC (Reliable Connected), 108
RD (Reliable Datagram), 108
RDMA (Remote Direct Memory Access), 175
advantages of, 104–106
architecture of, 106–107
control plane acceleration, 117–118

data security, 118–119
deployments, 110–112
history of, 103
operations, 108–109
RDMA over Converged Ethernet (RoCE), 19, 109–110
remote nonvolatile memory access, support for, 118
RoCE (RDMA over Converged Ethernet), 109–110
RoCEv2 and lossy networks, 112–117
scalability, 109
transport services, 108

Read/Write operations (RDMA), 108
Receive Queue (RQ), 106–107
Receive Side Coalescing (RSC), 150
Receive Side Scaling (RSS), 150
reconciliation, 215
Redis, 212–213
redundancy, network, 214
Redundant Array of Independent Disks (RAID), 126
registration, memory, 107
Reliable Connected (RC), 108
Reliable Datagram (RD), 108
Remote Direct Memory Access. *See* RDMA (Remote Direct Memory Access)
remote nonvolatile memory access, support for, 118
Representational State Transfer (REST) API, 54, 160, 208
Requests for Comment. *See* RFCs (Requests For Comment)
REST (Representational State Transfer) API, 54, 160, 208
RFCs (Requests For Comment), 11
RFC 1191, 22
RFC 1701, 18–19
RFC 1853, 18
RFC 1918, 79
RFC 1981, 22
RFC 2473, 18
RFC 2663, 79
RFC 4821, 22
RFC 7348, 20
RIB (routing information base), 68
Rivest-Shamir-Adleman (RSA), 89
RoCE (RDMA over Converged Ethernet), 19, 109–110
Rocket, 48
ROTPK (root of trust public key), 97
routing
centralized, 66
defined, 11
L2 forwarding, 12
L3 forwarding, 12–13
LPM (longest prefix match) forwarding, 13–14
SR (segment routing), 23–24
VRF (virtual routing and forwarding), 14
routing information base (RIB), 68
RPC framework, 208
RQ (Receive Queue), 106–107
RSA (Rivest-Shamir-Adleman), 89
RSC (Receive Side Coalescing), 150
RSS (Receive Side Scaling), 150
RTE Flow Filtering (DPDK), 74
runtimes, container, 48
Rupp, Karl, 131
Rust, 208

S

scalability/scaling
Dennard scaling, 4, 134–135
distributed management control plane, 212–214, 223–226
RDMA (Remote Direct Memory Access), 109
SCSI (Small Computer System Interface), 119
SCSI RDMA Protocol (SRP), 121
SDN (software-defined networking), 6
DPDK (Data Plane Development Kit), 68
gRIBI (gRBC Routing Information Base Interface), 67–68
OpenFlow, 64–66
overview of, 63–64
SD-WAN (software-defined wide-area network), 66–67

SDSPs (software defined services platforms)
distributed multiple, 220
federation of multiple, 220–223
single SDSP scaling, 219
SD-WAN (software-defined wide-area network), 66–67
Secure Production Identity Framework for Everyone (SPIFFE), 210
Secure Sockets Layer (SSL), 88, 93
security
APIs (application programming interfaces), 216–217
asymmetric encryption, 89–90
digital certificates, 90
distributed firewalls, 85–86
distributed management control plane, 210–211
encryption, 125
hashing, 90, 126
microsegmentation, 86–87
overview of, 84–85
PUF (physical unclonable function), 91
RDMA (Remote Direct Memory Access), 118–119
secure boot, 97
secure key storage, 90–91
secure tunnels, 22–23, 92
with DTLS (Datagram TLS), 94
with IPsec, 92–93
with TLS (Transport Layer Security), 93–94
security threats, 84–85
SPIFFE (Secure Production Identity Framework for Everyone), 210
SSL (Secure Sockets Layer), 88, 93
storage, 125
symmetric encryption, 89
TCP/TLS/HTTP implementation, 91
TLS (Transport Layer Security), 87–89, 91, 93–94, 210
VPNs (virtual private networks), 94–96
zero-trust security, 87
segment routing (SR), 23–24
selective retransmission, 115–117
Send Queue (SQ), 106–107
Send/Receive operations (RDMA), 108
Serializer/Deserializer (SerDes), 177
server buses, 143–144
service chaining, 25. *See also* tromboning
service mesh, 6–7
service providers, 164
services. *See also individual services*
overview of, 6
service mesh, 6–7
SHA hashing algorithms, 90, 126
Shacham, O., 131
shortest path first (SPF), 64
silicon design
7 nm process, 178
16 nm process, 177–178
"silicon sweet-spot," 176–177
Simple Network Management Protocol (SNMP), 64, 81
Single Root Input/Output Virtualization (SR-IOV), 45–46, 149–153, 175
single-thread CPU performance, 137–138
site-to-site VPNs (virtual private networks), 96
SKB (socket buffer), 77
Small Computer System Interface (SCSI), 119
SmartNIC, 7–8, 154–155
Snappy, 125
SNIA (Storage Networking Industry Association), 127
SNMP (Simple Network Management Protocol), 64, 81
SoC (System on a Chip), 177
socket buffer (SKB), 77
software, distributed services platforms in, 164–166
software defined services platforms. *See* SDSPs (software defined services platforms)
software-defined networking. *See* SDN (software-defined networking)
software-defined wide-area network (SD-WAN), 66–67
solid state drives (SSDs), 119–120
source MAC address field (Ethernet packets), 12
SPAN (switched port analyzer) ports, 192
SPEC Integer benchmark, 138
Spectre, 84, 90–91

speed, distributed services platforms, 158
SPF (shortest path first), 64
SPIFFE (Secure Production Identity Framework for Everyone), 210
SQ (Send Queue), 106–107
SR (segment routing), 23–24
SR-IOV (Single Root Input/Output Virtualization), 45–46, 149–153, 175
SRP (SCSI RDMA Protocol), 121
SSDs (solid state drives), 119–120
SSL (Secure Sockets Layer), 88, 93
Starovoitov, Alexei, 76
stateful NAT (network address translation), 79
stateless offloads, 150
static bonding, 40
storage
data plane model of storage protocols, 120–122
efficiency of, 125–126
NVMe-oF (NVME over Fabrics), 120
offloading and distributing, 126–127
persistent memory as, 127
reliability of, 126
SCSI (Small Computer System Interface), 119
security, 125
SSDs (solid state drives), 119–120
storage services by type, 124
virtualization and, 122–124
Storage Networking Industry Association (SNIA), 127
Sun, 103
Swagger, 54, 160
Swift module (OpenStack), 56
switched port analyzer (SPAN) ports, 192
switches. *See also* bridging and routing
distributed services platforms in, 169–170
ToR (top of rack), 168
vSwitches
BPF (Berkeley Packet Filter), 76
classification of, 69
DPDK Generic Flow API, 74
eBPF (extended Berkeley Packet Filter), 76
overview of, 39–40
OVS (Open vSwitch), 70–73
summary of, 78–79
tc-flower, 73–74
VPP (Vector Packet Processing), 75
XDP (eXpress Data Path), 76–77
symmetric encryption, 89
synchronization, time, 152
System on a Chip (SoC), 177

T

Tandem Computers, 103
target users, determining, 163–164
TCAM (ternary content-addressable memory), 13, 139
tc-flower, 73–74
TCP (Transmission Control Protocol), 19, 91
TCP segmentation offload (TSO), 150
telecommunications, 57
telemetry, 8, 80–81
ternary content-addressable memory (TCAM), 13, 139
ternary matches, 13–14
TICK, 212
time series databases (TSDBs), 214
Time Server, 29
time synchronization, 152
time to live (TTL), 12
TLS (Transport Layer Security), 87–89, 91, 93–94, 210
top of rack (ToR) switches, 14, 168
topologies, Clos, 14–15
ToR (top of rack) switches, 14, 168
Torvolds, Linus, 96
TPM (Trusted Platform Module), 210
traffic shaping, 150
transactional semantics, 217
transistor count
Amdahl's Law, 4
Dennard scaling, 4, 134–135
historical perspective of, 130–131
historical trend data, 131–132
Moore's law, 4, 103, 132–134, 138
single-thread performance, 137–138
technical factors limiting, 136–137

Transmission Control Protocol (TCP), 19, 91
Transport Layer Security (TLS), 87–89, 91, 93–94, 210
transport services, 108
tromboning, 5, 157
example of, 25
hybrid, 27
with VRF (virtual routing and forwarding), 26–27
with VXLAN (Virtual Extensible LAN), 25–26
troubleshooting
distributed management control plane, 209
distributed services platforms, 159–160
networking services, 80–81
Trusted Platform Module (TPM), 210
TSDBs (time series databases), 214
TSO (TCP segmentation offload), 150
TTL (time to live), 12
tunnels. *See also* encapsulation
defined, 17
endpoints, 17
secure, 22–23, 92
with DTLS (Datagram TLS), 94
with IPsec, 92–93
with TLS (Transport Layer Security), 93–94

U

UC (Unreliable Connected), 108
UD (Unreliable Datagram), 108
UDP (User Datagram Protocol), 19
underlay networks, 16
unified fabric, support for, 151
unified networks
hosts with, 100–102
software stack, 101–102
University of Cambridge Computer Laboratory, XEN, 46
Unreliable Connected (UC), 108
Unreliable Datagram (UD), 108
upstream, 72
User Datagram Protocol (UDP), 19
users, target, 163–164
utilities. *See individual utilities*

V

Vahdat, Amin, 66
vCenter (VMware), 41
Vector Packet Processing (VPP), 75
Verilog, 182
VERSION API (CNI), 50
vETh (virtual ethernet) ports, 39, 65
VFP (Virtual Filtering Platform), 29–30
VFs (virtual functions), 152, 175
VHDL, 182
VIA (Virtual Interface Architecture), 103
VID (VLAN identifier), 12, 19–20
VirtIO (Virtual I/O), 44–46, 153–154
virtual ethernet (vEth) ports, 39, 65
Virtual Extensible LAN. *See* VXLAN (Virtual Extensible LAN)
Virtual Filtering Platform (VFP), 29–30
virtual functions (VFs), 152, 175
Virtual Interface Architecture (VIA), 103
virtual machines (VMs), 37–40
Virtual Network ID (VNID), 21
virtual private networks (VPNs), 94–96
virtual routing and forwarding. *See* VRF (virtual routing and forwarding)
virtualization. *See also* networking services
advantages of, 34–35
clouds and, 35–37
containers
CNI (Container Network Interface), 49–50
container runtimes, 48
CSI (Container Storage Interface), 50
Kata Containers, 49
Kubernetes, 50–52
LXC (LinuX Containers), 47
overview of, 47–48
full, 39
hypervisors
Hyper-V, 41–43
KVM (Kernel-based Virtual Machine), 36, 43–46
overview of, 37–40
QEMU (Quick EMUlator), 43
types of, 38–39

VMware ESXi, 40–41
XEN, 46
libvirt, 45
microservice architecture
gRPC, 54–55
overview of, 52–54
REST API, 54
NFV (Network Function Virtualization), 57
OpenStack, 55–57
paravirtualization, 39
remote storage and, 122–124
SR-IOV (Single Root Input/Output Virtualization), 149–153
vETh (virtual ethernet) ports, 39, 65
VFP (Virtual Filtering Platform), 29–30
VFs (virtual functions), 152, 175
VIA (Virtual Interface Architecture), 103
VirtIO (Virtual I/O), 44–46, 153–154
virtualized versus bare-metal servers, 161
VLAN tagging support, 151
VMs (virtual machines), 37–40
VRF (virtual routing and forwarding), 14, 26–27, 29–30
vSwitches
BPF (Berkeley Packet Filter), 76
classification of, 69
DPDK Generic Flow API, 74
eBPF (extended Berkeley Packet Filter), 76
overview of, 39–40
OVS (Open vSwitch), 70–73
summary of, 78–79
tc-flower, 73–74
VPP (Vector Packet Processing), 75
XDP (eXpress Data Path), 76–77
VXLAN (Virtual Extensible LAN)
encapsulation, 19–22
tromboning with, 25–26

VLAN identifier (VID), 12, 19–20
VLAN tagging support, 151
VMs (virtual machines), 37–40. *See also* hypervisors
VMware
ESXi, 40–41
NSX, 41, 164
vCenter, 41
vSAN, 41
VNID (Virtual Network ID), 21
VPNs (virtual private networks), 94–96
VPP (Vector Packet Processing), 75
VRF (virtual routing and forwarding), 14, 26–27, 191
vsAN (VMware), 41
vSwitches
BPF (Berkeley Packet Filter), 76
classification of, 69
DPDK Generic Flow API, 74
eBPF (extended Berkeley Packet Filter), 76
overview of, 39–40
OVS (Open vSwitch), 70–73
summary of, 78–79
tc-flower, 73–74
VPP (Vector Packet Processing), 75
XDP (eXpress Data Path), 76–77
VXLAN (Virtual Extensible LAN)
encapsulation, 19–22
tromboning with, 25–26

W

Wireguard, 96
WRs (work requests), 106–107

X

X.509 certificates, 210
XDP (eXpress Data Path), 76–77
XEN, 46
xenproject.org, 46
XenSource, Inc., 46
XTS encryption mode, 89

Y-Z

YANG (Yet Another Next Generation), 54, 160
zero-copy, 104, 105
zero-trust security, 87
Zombieland, 90–91
Zookeeper, 212–213