A Functional Guide to the NIC Evolution

Hendrich Hernandez
AT THE NETWORK’S EDGE

A Functional Guide to the NIC Evolution

Hendrich Hernandez
© 2017 Dell Inc. or its subsidiaries. All Rights Reserved. Dell, EMC, Dell EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be trademarks of their respective owners.

The Author asserts the moral right to be identified as the author of this work.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior permission from the author or publisher.


This paperback edition is published in 2017 by

Dell Inc.

1 Dell Way Round Rock TX, U.S.A.

Dell.com

Printed in the U.S.A.
# Table of Contents

Author ........................................................................................................ xvi
Acknowledgments ...................................................................................... xvii
Audience ..................................................................................................... xviii
Preface .......................................................................................................... xix

## Chapter 1
Introduction ............................................................................................... 1

## Chapter 2
System Architecture .................................................................................... 5
  Central Processing Unit ........................................................................... 7
  Chipset ....................................................................................................... 7
  Memory ....................................................................................................... 7
  Peripheral Component Interconnect Express ............................................ 7
  Baseboard Management Controller ....................................................... 9
  Network Interface Card .......................................................................... 10

## Chapter 3
NIC Form Factors ....................................................................................... 13
  Server Interconnectivity ......................................................................... 13
  Physical Classification ........................................................................... 14
    PCIe Adapter ....................................................................................... 15
    LAN on Motherboard ......................................................................... 15
    LOM Riser Card ................................................................................. 16
    Network Daughter Card .................................................................... 16
    Mezzanine ......................................................................................... 17
Virtual Classification .................................................. 18
  vNIC ............................................................... 18
  VLAN ............................................................... 18
  VxLAN .............................................................. 18
  SR-IOV VFs .......................................................... 19

Functional Classification .............................................. 19
  Network Interface Card ............................................ 19
  Converged Network Adapter ...................................... 19
  Host Bus Adapter .................................................. 19
  Host Channel Adapter ............................................. 19
  Host Fabric Interface ............................................... 20

Chapter 4

Network Communications ............................................. 21
  Host Perspective ................................................... 21
    Memory Buffers .................................................. 23
    Filters ............................................................ 24
    Queues ........................................................... 24
  Direct Memory Access ............................................. 25
  Interrupts .......................................................... 25
  Control Interfaces .................................................. 25
    NC-SI ............................................................. 25
    SMBus ............................................................ 26
    I2C ............................................................... 26
  Network Perspective ............................................... 26
    Packet Buffers .................................................. 28
    Spanning Tree Algorithm ....................................... 28
  Performance ........................................................ 30
    Theoretical Performance ......................................... 30

Chapter 5

PCIe Host Interface .................................................. 31
  PCIe Identification .................................................. 31
  PCIe Link .......................................................... 31
    Physical Layer .................................................... 32
    Initialization ..................................................... 32
    Enumeration ..................................................... 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Mapping</td>
<td>32</td>
</tr>
<tr>
<td>Transactions</td>
<td>33</td>
</tr>
<tr>
<td>Writes</td>
<td>33</td>
</tr>
<tr>
<td>Reads</td>
<td>33</td>
</tr>
<tr>
<td>Clocks</td>
<td>34</td>
</tr>
<tr>
<td>Power</td>
<td>34</td>
</tr>
<tr>
<td>Flow Control</td>
<td>34</td>
</tr>
<tr>
<td>MSI/MSI-X</td>
<td>34</td>
</tr>
<tr>
<td>Power Management</td>
<td>34</td>
</tr>
<tr>
<td><strong>Chapter 6</strong></td>
<td></td>
</tr>
<tr>
<td>Ethernet Technologies</td>
<td>35</td>
</tr>
<tr>
<td>GE</td>
<td>36</td>
</tr>
<tr>
<td>10GE</td>
<td>37</td>
</tr>
<tr>
<td>10GBase-T PHY</td>
<td>40</td>
</tr>
<tr>
<td>25GE</td>
<td>40</td>
</tr>
<tr>
<td>25G/40GBASE-T</td>
<td>42</td>
</tr>
<tr>
<td>40GE/100GE</td>
<td>42</td>
</tr>
<tr>
<td>400GE</td>
<td>43</td>
</tr>
<tr>
<td><strong>Chapter 7</strong></td>
<td></td>
</tr>
<tr>
<td>Ethernet Link</td>
<td>45</td>
</tr>
<tr>
<td>Duplex</td>
<td>45</td>
</tr>
<tr>
<td>Wake on LAN</td>
<td>45</td>
</tr>
<tr>
<td>Auto-negotiation</td>
<td>45</td>
</tr>
<tr>
<td>10BaseT Link Test Pulses</td>
<td>47</td>
</tr>
<tr>
<td>100BaseT/1000BaseT Fast Link Pulses</td>
<td>48</td>
</tr>
<tr>
<td>FLP Bursts</td>
<td>49</td>
</tr>
<tr>
<td>FLP Burst Encoding</td>
<td>49</td>
</tr>
<tr>
<td>Base Link Code Word</td>
<td>50</td>
</tr>
<tr>
<td>Technology Ability Field</td>
<td>50</td>
</tr>
<tr>
<td>Gigabit Auto-negotiation</td>
<td>51</td>
</tr>
<tr>
<td>Parallel Detection</td>
<td>51</td>
</tr>
<tr>
<td>Duplex Mismatching</td>
<td>52</td>
</tr>
<tr>
<td>Enabling Migration to Gigabit Speeds</td>
<td>52</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Maximum Transmission Unit</td>
<td>52</td>
</tr>
<tr>
<td>Flow Control</td>
<td>53</td>
</tr>
<tr>
<td>Data Center Bridging</td>
<td>53</td>
</tr>
<tr>
<td><strong>Chapter 8</strong></td>
<td></td>
</tr>
<tr>
<td>MAC/PHY</td>
<td>55</td>
</tr>
<tr>
<td>Media Access Controller</td>
<td>55</td>
</tr>
<tr>
<td>Interrupt Moderation</td>
<td>56</td>
</tr>
<tr>
<td>Wake on LAN</td>
<td>56</td>
</tr>
<tr>
<td>Direct Memory Access</td>
<td>56</td>
</tr>
<tr>
<td>Precision Timing Protocol</td>
<td>57</td>
</tr>
<tr>
<td>Physical Layer</td>
<td>57</td>
</tr>
<tr>
<td>Internal PHY</td>
<td>58</td>
</tr>
<tr>
<td>External PHY</td>
<td>58</td>
</tr>
<tr>
<td>PHY Features</td>
<td>58</td>
</tr>
<tr>
<td>Auto-Negotiation</td>
<td>58</td>
</tr>
<tr>
<td>Forward Error Correction</td>
<td>58</td>
</tr>
<tr>
<td>Energy Efficient Ethernet</td>
<td>59</td>
</tr>
<tr>
<td>Physical Communication Channel</td>
<td>61</td>
</tr>
<tr>
<td>Copper Channels</td>
<td>62</td>
</tr>
<tr>
<td>Optical Channels</td>
<td>63</td>
</tr>
<tr>
<td>Connector Types</td>
<td>64</td>
</tr>
<tr>
<td>Channel Types</td>
<td>64</td>
</tr>
<tr>
<td>CAT5/6 UTP</td>
<td>64</td>
</tr>
<tr>
<td>CAT7 STP</td>
<td>65</td>
</tr>
<tr>
<td>DAC</td>
<td>65</td>
</tr>
<tr>
<td>Fiber</td>
<td>66</td>
</tr>
<tr>
<td>Backplane/Midplane</td>
<td>66</td>
</tr>
<tr>
<td><strong>Chapter 9</strong></td>
<td></td>
</tr>
<tr>
<td>NIC BIOS</td>
<td>67</td>
</tr>
<tr>
<td>Option ROM</td>
<td>67</td>
</tr>
<tr>
<td>PXE Boot</td>
<td>68</td>
</tr>
<tr>
<td>iSCSI Boot</td>
<td>69</td>
</tr>
<tr>
<td>Fibre Channel over Ethernet Boot</td>
<td>70</td>
</tr>
<tr>
<td>UEFI Boot</td>
<td>71</td>
</tr>
</tbody>
</table>
Chapter 10
NIC Software Components ................................................. 73
Base Driver ........................................................................ 73
NIC Teaming ...................................................................... 74
Network Addressing .......................................................... 76
Teaming and Network Addresses ........................................ 76
Link Status ......................................................................... 77
Description of Teaming Modes .......................................... 77
OS Based Teaming ............................................................. 81
Virtual LANs .................................................................... 82

Chapter 11
NIC Stateless Offloads ....................................................... 85
Checksum Offload .............................................................. 85
Large send offload ............................................................ 86
Large Receive Offload/Receive Side Coalescing .................... 86
Receive Side Scaling .......................................................... 86
Virtual RSS ....................................................................... 87
Virtual Machine Queues ..................................................... 87
VxLAN .............................................................................. 88
VxLAN Packet Offloads ...................................................... 89
NVGRE ............................................................................. 89
NVGRE Packet Offloads ..................................................... 90

Chapter 12
I/O Virtualization .......................................................... 93
Emulated I/O ..................................................................... 94
SR-IOV ............................................................................ 95
SR-IOV With Internal Switching ......................................... 95
SR-IOV With External Switching ....................................... 96
Host Platform Requirements .............................................. 98
NIC Partitioning ............................................................... 99
Use Cases ......................................................................... 100
NIC Partitioning Benefits ................................................ 100
NPar vs. SR-IOV .............................................................. 101
Network Function Virtualization ....................................... 103
# Table of Contents

## Chapter 13

- iSCSI .............................................................. 105
  - Protocol Stack .................................................. 106
  - Connection ....................................................... 107
  - Session .......................................................... 107
  - Protocol Data Unit ............................................ 108
  - iSCSI HBA ....................................................... 108

## Chapter 14

- Fibre Channel over Ethernet ..................................... 109
  - FCoE Protocol ................................................... 110
  - FCoE Addressing ................................................ 110
  - Discovery ......................................................... 111
  - Encapsulation ................................................... 111
  - FCoE Frame ....................................................... 112
  - FCoE Initialization Protocol .................................. 113
  - Login .............................................................. 113
  - FCoE Adapter ..................................................... 114
  - Offload Architecture .......................................... 114
  - Fibre Channel Protocol Offload .............................. 114
  - FCoE Attributes ................................................ 115
  - FCoE Topologies ................................................ 116
    - Non-Converged Topology .................................... 116
    - Server I/O Consolidation .................................... 116
    - Multi-FCF, Single FCF Fabric- Multi-hop DCB .......... 117
    - Native FCoE Targets ......................................... 117
  - Fibre Channel Backbone - 6 (FC-BB-6) .................... 118

## Chapter 15

- Data Center Bridging ............................................ 119
  - Legacy Ethernet Frame Format ............................... 120
  - IEEE 802.1p Packet Priority ................................ 120
  - Legacy Flow Control .......................................... 121
  - Priority-based Flow Control .................................. 122
    - Flow Control Frame ......................................... 123
  - Enhanced Transmission Selection ........................... 123
Data Center Bridging Exchange Protocol .................................................. 124
ETS TLVs .................................................................................. 126
PFC TLVs .................................................................................. 126
Application Priority TLV .............................................................. 126

Chapter 16
Low Latency .................................................................................. 127
Key Factors ................................................................................ 128
Applications ............................................................................... 128
SMB Direct ............................................................................... 128
Low Latency Protocols ................................................................. 129
  Data Plane Development Kit (DPDK)/Packet Direct .................. 129
  PacketDirect Provider Interface ............................................. 130
  RDMA ................................................................................... 130
OpenOnload ............................................................................. 137

Chapter 17
Security ...................................................................................... 139
Internet Protocol Security ........................................................... 140
  IPSec Components .................................................................... 140
  The ESP Header ....................................................................... 141
  AH Protocol ............................................................................. 142
  Authenticating IPSec ................................................................. 144
  IPSec Security Associations ..................................................... 146
  IPSec Security Policies ............................................................. 147
  IPSec Modes ............................................................................ 147
SSL/TLS ..................................................................................... 150
MACSec .................................................................................... 150
Protocols Comparison ................................................................. 152

Chapter 18
Stateful Offloads ........................................................................ 153
Network .................................................................................... 154
  TOE ..................................................................................... 154
  TCP Chimney ......................................................................... 155
  IPSec .................................................................................... 155
Storage ................................................................. 156
iSOE ................................................................. 156
FCoE ................................................................. 156
MPI ................................................................ 156
RoCEv1 .............................................................. 156
RoCEv2 .............................................................. 157
iWARP ............................................................... 158

The Future ............................................................. 159

Appendix A
Glossary .............................................................. 161

Appendix B
References ............................................................ 171

Appendix C
Works Cited .......................................................... 175

Index ................................................................ 181
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NIC Environmental Components</td>
<td>xix</td>
</tr>
<tr>
<td>2</td>
<td>NIC Functional Space</td>
<td>xx</td>
</tr>
<tr>
<td>3</td>
<td>NIC Operational Space</td>
<td>xxi</td>
</tr>
<tr>
<td>4</td>
<td>Physical Server Edge</td>
<td>xxi</td>
</tr>
<tr>
<td>5</td>
<td>Virtualized Server Edge</td>
<td>xxii</td>
</tr>
<tr>
<td>6</td>
<td>OSI Model</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>PCIe Connectors</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Simplified System Block Diagram</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>The Network Edge</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>NC-SI pass-thru</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>NIC Block Diagram</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>NIC Characteristics</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>PCIe Adapters</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>Rack NDC</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>Blade NDC</td>
<td>17</td>
</tr>
<tr>
<td>16</td>
<td>Blade Mezzanine</td>
<td>17</td>
</tr>
<tr>
<td>17</td>
<td>Packet Flow</td>
<td>22</td>
</tr>
<tr>
<td>18</td>
<td>Memory Copies</td>
<td>23</td>
</tr>
<tr>
<td>19</td>
<td>Hardware Arbitration</td>
<td>26</td>
</tr>
<tr>
<td>20</td>
<td>STP</td>
<td>29</td>
</tr>
<tr>
<td>21</td>
<td>10GE</td>
<td>37</td>
</tr>
<tr>
<td>22</td>
<td>10GE Applications</td>
<td>39</td>
</tr>
<tr>
<td>Figure 23.</td>
<td>PHY Layer Blocks</td>
<td>40</td>
</tr>
<tr>
<td>Figure 24.</td>
<td>25GE Layers</td>
<td>42</td>
</tr>
<tr>
<td>Figure 25.</td>
<td>100GE and 40GE Choices</td>
<td>43</td>
</tr>
<tr>
<td>Figure 26.</td>
<td>Layer Model (Redrawn from the IEEE Std 802.3, 1998 Edition)</td>
<td>47</td>
</tr>
<tr>
<td>Figure 27.</td>
<td>FLP and NLP Comparison</td>
<td>48</td>
</tr>
<tr>
<td>Figure 28.</td>
<td>FLP Burst Timing</td>
<td>49</td>
</tr>
<tr>
<td>Figure 29.</td>
<td>FLP Burst Encoding</td>
<td>49</td>
</tr>
<tr>
<td>Figure 30.</td>
<td>Base Link Code Word Bit Positions</td>
<td>50</td>
</tr>
<tr>
<td>Figure 31.</td>
<td>NIC Block Diagram</td>
<td>55</td>
</tr>
<tr>
<td>Figure 32.</td>
<td>NIC DMA</td>
<td>56</td>
</tr>
<tr>
<td>Figure 33.</td>
<td>EEE States</td>
<td>59</td>
</tr>
<tr>
<td>Figure 34.</td>
<td>EEE in the Stack</td>
<td>60</td>
</tr>
<tr>
<td>Figure 35.</td>
<td>EEE Operation</td>
<td>60</td>
</tr>
<tr>
<td>Figure 36.</td>
<td>Link Speed Transitions</td>
<td>61</td>
</tr>
<tr>
<td>Figure 37.</td>
<td>Supported Cables</td>
<td>62</td>
</tr>
<tr>
<td>Figure 38.</td>
<td>Cable Length</td>
<td>63</td>
</tr>
<tr>
<td>Figure 39.</td>
<td>UTP Cable</td>
<td>64</td>
</tr>
<tr>
<td>Figure 40.</td>
<td>RJ-45 Connector</td>
<td>65</td>
</tr>
<tr>
<td>Figure 41.</td>
<td>STP Cable</td>
<td>65</td>
</tr>
<tr>
<td>Figure 42.</td>
<td>Direct Attach Copper Cable</td>
<td>65</td>
</tr>
<tr>
<td>Figure 43.</td>
<td>Fiber Optic Cable</td>
<td>66</td>
</tr>
<tr>
<td>Figure 44.</td>
<td>LC Connector</td>
<td>66</td>
</tr>
<tr>
<td>Figure 45.</td>
<td>PXE Boot</td>
<td>68</td>
</tr>
<tr>
<td>Figure 46.</td>
<td>iSCSI Boot Process</td>
<td>70</td>
</tr>
<tr>
<td>Figure 47.</td>
<td>Software Stack</td>
<td>73</td>
</tr>
<tr>
<td>Figure 48.</td>
<td>VLANs</td>
<td>82</td>
</tr>
<tr>
<td>Figure 49.</td>
<td>VLANs Example</td>
<td>83</td>
</tr>
<tr>
<td>Figure 50.</td>
<td>VMQ</td>
<td>88</td>
</tr>
<tr>
<td>Figure 51.</td>
<td>VxLAN Header</td>
<td>89</td>
</tr>
<tr>
<td>Figure 52.</td>
<td>NVGRE</td>
<td>90</td>
</tr>
<tr>
<td>Figure 53.</td>
<td>Virtualization Block Diagram</td>
<td>94</td>
</tr>
</tbody>
</table>
Figure 54. Software Emulation ........................................... 95
Figure 55. SR-IOV With VEB ........................................... 96
Figure 56. SR-IOV With External Switching ......................... 97
Figure 57. NPar ............................................................. 101
Figure 58. NIC Partitioning Example .................................. 102
Figure 59. Protocol Stacks ............................................... 105
Figure 60. iSCSI Protocol Layers ..................................... 106
Figure 61. iSCSI Software Stacks ..................................... 106
Figure 62. iSCSI Communications .................................... 107
Figure 63. Storage Protocol Stack .................................... 110
Figure 64. FCoE Frame .................................................. 112
Figure 65. FIP Header .................................................... 113
Figure 66. Initialization ................................................... 113
Figure 67. FCoE CNA Block Diagram ............................... 114
Figure 68. Multiple I/O Fabric Connectivity .......................... 116
Figure 69. Single FCF .................................................... 116
Figure 70. DCB in the DC .............................................. 117
Figure 71. FCoE Targets ................................................ 117
Figure 72. Frame Format ................................................. 120
Figure 73. Priority Tag ................................................... 120
Figure 74. Flow Control ................................................. 121
Figure 75. Priority-based Pause ....................................... 122
Figure 76. PFC Frame .................................................... 123
Figure 77. Typical Network Stack ..................................... 127
Figure 78. SMB Direct ................................................... 129
Figure 79. Buffer Copies ............................................... 131
Figure 80. RDMA Concept ............................................. 132
Figure 81. IB vs. RoCE ................................................ 132
Figure 82. RoCE Headers .............................................. 133
Figure 83. iWARP Stack ................................................. 134
Figure 84. iWARP NIC ............................................... 134
Figure 85. iWARP Header .............................................136
Figure 86. iWARP Block Diagram ........................................136
Figure 87. SSL .................................................................139
Figure 88. IPSec ESP Packet .............................................141
Figure 89. 3DES Process ..................................................141
Figure 90. AH Packet .........................................................142
Figure 91. Authentication Header ..........................................143
Figure 92. Hashing ..............................................................143
Figure 93. Security Association ............................................146
Figure 94. Transport Mode ................................................148
Figure 95. Tunnel Mode ......................................................148
Figure 96. Multiple Tunnels ...............................................149
Figure 97. Received Packet ................................................149
Figure 98. MACSec vs. IPSec .............................................151
Figure 99. MACSec Header ................................................151
Figure 100. Protocol Stacks .................................................153
Figure 101. TOE ...............................................................154
Figure 102. NIC vs. TOE .....................................................155
Figure 103. RoCE and IB .....................................................157
Figure 104. RoCE and RoCEv2 Frame ....................................157
Figure 105. iWARP NIC ......................................................158
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>PCIe Bandwidth</td>
<td>8</td>
</tr>
<tr>
<td>Table 2</td>
<td>PCIe Device Bandwidth Coverage</td>
<td>8</td>
</tr>
<tr>
<td>Table 3</td>
<td>NIC BOM</td>
<td>11</td>
</tr>
<tr>
<td>Table 4</td>
<td>Device Functionality</td>
<td>20</td>
</tr>
<tr>
<td>Table 5</td>
<td>Packet Rates</td>
<td>30</td>
</tr>
<tr>
<td>Table 6</td>
<td>NIC Speeds</td>
<td>36</td>
</tr>
<tr>
<td>Table 7</td>
<td>Ethernet Layers</td>
<td>38</td>
</tr>
<tr>
<td>Table 8</td>
<td>10GE PMD</td>
<td>38</td>
</tr>
<tr>
<td>Table 9</td>
<td>10GE Fiber Media</td>
<td>39</td>
</tr>
<tr>
<td>Table 10</td>
<td>Priority Resolution</td>
<td>51</td>
</tr>
<tr>
<td>Table 11</td>
<td>DCB Features</td>
<td>54</td>
</tr>
<tr>
<td>Table 12</td>
<td>Connector Types</td>
<td>64</td>
</tr>
<tr>
<td>Table 13</td>
<td>I/O Virtualization Architectures</td>
<td>93</td>
</tr>
<tr>
<td>Table 14</td>
<td>NPar vs. SR-IOV</td>
<td>102</td>
</tr>
<tr>
<td>Table 15</td>
<td>DCB Protocol Features</td>
<td>125</td>
</tr>
<tr>
<td>Table 16</td>
<td>Security Techniques</td>
<td>151</td>
</tr>
<tr>
<td>Table 17</td>
<td>IPsec vs. SSL</td>
<td>152</td>
</tr>
</tbody>
</table>
Hendrich (Rich) Hernandez is a Distinguished Engineer at Dell EMC. He has over 25 years of experience in networking in general and about 17 years of experience in server networking in particular. He has filed 32 patents with the U.S. Patent and Trademark Office as of the writing of this book. Rich is in a unique position to understand and appreciate the evolution of the NIC over the last 17 years. He is able to explain the many aspects of host based networking, including the hardware and software components and features.
First and foremost, I am grateful to my wife, Mirta, and daughters, Karem, Velvette, Jackie, and Amy, for making it easy to work on this project during many weekends.

I want to thank Lee Ballard and Neal Beard for conducting technical reviews and providing many suggestions to improve the book.

I want to recognize that a lot of the knowledge I have included in this book came from the experience gained at Dell EMC over the last 17 years. I have worked with the best in the data communications field and collaborated with our partners to bring state-of-the-art products and solutions to our customers.

I want to give a special thanks to the following team members who provided graphics design and technical editing:

- Trini Thomas
- Ramji K
- Thomas Mathew
- Steve Bracamontez
- Doug Kaiser

I want to recognize the support from our management team. In particular, a special thanks to the following for their encouragement and continued support:

- Jim Leftwich
- Jerele Neeld
- Keith Winkeler
- Jaime Edwards
- Bryan Kelly
- Thomas Garvens
- Kurtis Bowman
Readers of this book should have a basic understanding of computer systems and networking. This book is meant to be a functional reference that describes the many capabilities and features of the NIC. It is targeted for onboarding anyone new to server networking or anyone seeking a broader understanding of server networking. This includes customers and administrators of computer and networking systems. This book will also be useful to validation/test operations, marketing, sales and support personnel.
I decided to write this book to provide a deeper understanding of the Network Interface Card (NIC). In the last couple of years, it became clear to me that the host (server) network interface is a complex and ever-changing operating environment. This environment includes a nexus of the operating system (OS) networking stack, the PCI Express (PCIe) host interface, and the local area network (LAN). While there are many books that treat each one separately, I saw the need to write a book that covered each of these topics in one place. This will help readers gain a deeper understanding of the NIC environment.

Figure 1. NIC Environmental Components
Another reason for writing a book on the server NIC is the richness of functionality that I have seen designed in the NIC over the last 20 years. This book explains the operational concepts of the various NIC features and protocols, based on my experience at Dell EMC.

Over the years, I have received many questions on how the NIC operates along the vectors shown in Figure 2. These questions were about software, firmware, and hardware components. The explanations for these questions became the basis of this book.

This book describes the server NIC that resides at the edge of the LAN between the end node and the network access layer. The NIC is a seemingly small and simple component of the overall information system. However, without this device to bridge the compute, storage, and network domains, all the advances in processing, memory, and storage cannot be accessed by the end users’ applications. The NIC is a necessary component that enables other advances in computing and networking.

The NIC architecture includes the data, control, and management planes. The data plane is implemented in the hardware as the fast path. The control plane is the slow path implemented via a driver and firmware running on embedded cores in the NIC. The management plane can be implemented in-band (CIM, SNMP) over the Ethernet link of the NIC or out-of-band via sideband interfaces such as Network Controller Sideband Interface (NC-SI) and System Management Bus (SMBus).
The NIC is primarily a communication interface device between a server and the network. It translates the I/O interfaces (usually PCIe) in the server into the interfaces facing the network (usually Ethernet). This is a basic function of the NIC, but it is only one of the capabilities that can be brought to bear at this critical interface. The NIC is the edge device acting as a corridor between the compute domain and network domain.

**Figure 3.** NIC Operational Space

**Figure 4.** Physical Server Edge
The advent of virtualization moved the edge of the network inside the server. The virtual switch was added to provide for east-west communications between VMs on the same host as shown in Figure 5. Virtual NICs are created in the software for use by the VMs. The vSwitch connects the vNICs to other vNICs or to the physical uplink.

![Figure 5. Virtualized Server Edge](image)

Therefore, considering the importance and complexity of the NIC, I felt a book dedicated to this subject would benefit the IT community.

This book describes the multitude of protocols and interfaces, both physical and logical, software infrastructure, offloads, features, and multiple personalities of the NIC. This book does not cover wireless NICs; it focuses only on physically connected interfaces over Ethernet. All references to IP apply to IPv4. IPv6 is not covered since it is a layer above the NIC in most cases.

While this book offers generic concepts, explanations, and abstractions, it is not an implementation design guide. Implementations will vary with product-specific offerings. The reader should consult the product documentation for specific details. This book should be used as a reference to understand the rich legacy and to ground oneself for the evolution that is surely coming to the world of the NIC.
Chapter 1

**Introduction**

The NIC is the subsystem responsible for connecting an end node to the network. An end node may be a computer system or a network storage device. More specifically, these network devices support platforms such as PCs, workstations, laptops, servers, and storage arrays. The NIC provides network access to these devices.

The NIC, at the most basic level, implements the physical and data link layers of the OSI model. The physical layer describes media specifications such as the electrical signal characteristics, signal encoding, and Digital Signal Processing (DSP) techniques to allow transmission over a supported communications channel. The supported channel can be a copper or optical cable including the required connectors. The PHY (Physical Layer) is the component of the NIC that implements the OSI physical layer and provides link training and auto-negotiation (AN) to set the link speed and other parameters. The data link layer comprehends the protocols used to frame the data payload for transmission over the physical channel. The most commonly used Layer 2 protocol is that defined by the IEEE Std 802.3-2015 Ethernet Specification. Figure 6 shows the OSI model.

![Figure 6. OSI Model](image-url)
Another function of the NIC is to act as an interface to the host end node. The most prevalent host interface on NICs is PCIe. This interface defines the electrical and mechanical specifications for NICs. The PCIe link on a NIC can be x1, x4, x8, or x16 lanes. PCIe version 3, the current implementation at the writing of this book, supports a data rate of 8 Gb/s per lane. The mechanical specifications include support for standard height/half-length cards, standard height/full-length cards, and low profile cards. The PCIe connectors are shown in Figure 7.

![PCIe Connectors](image)

**Figure 7.** PCIe Connectors

The NIC subsystem also includes driver software running on the host OS. The driver implements the software to control the behavior of the NIC for functions such as send and receive. The driver conforms to the OS API specifications such as Network Driver Interface Specification (NDIS) for Windows or NetDev for Linux.

In addition to drivers that run as part of the OS kernel (in most cases, as some drivers run in user space) and are executed on the host CPU, the NIC includes firmware that implements pre-OS functions such as network boot. This code is stored in electrically erasable programmable read-only memory (EEPROM) or flash, known as Option ROM. The boot code is loaded to the host memory and executed by the host CPU. In addition, the default values of the NIC registers are set.

Many modern NICs have embedded CPU cores to implement specific offloads in firmware. This firmware may be packaged and loaded to the NIC in various ways, but always with the purpose of offloading the host CPU from performing network operations. This frees up CPU cycles. Examples include TCP Offload Engine (TOE), iSCSI Offload Engine (iSOE), and Fibre Channel over Ethernet (FCoE). Other NICs perform offloads for Remote Direct Memory Access (RDMA) devices with iWARP and RDMA over Converged Ethernet (RoCE).

Once power is applied, the PCIe links initialize and train to adjust the desired link speed. System BIOS discovers and enumerates the PCIe hierarchy formed by the root complex (RC), switches, and end devices. PCIe configuration cycles set up
the NIC PCIe address space. Other pre-OS operations such as network boot (PXE, iSCSI, FCoE) take place via the code stored in Non-Volatile Random Access Memory (NVRAM) such as an EEPROM. This code enables the necessary software stack components to implement network communications during the boot process.

Once the OS is loaded onto the host memory and the network stack is initialized, the hardware is detected and drivers are loaded onto the host memory and the code is executed on the host CPU. The drivers implement the Application Programming Interface (API) function calls to set up the NIC for data communications over the network. The driver, via the API, interfaces with the TPC/IP stack for upper-layer protocol (ULP) communications on one side and the hardware abstraction layer (NDIS, etc.) on the other side.

The NIC has similar or equivalent components in other types of networks such as Host Bus Adapters (HBAs) for Fibre Channel (FC) and Host Controller Adapters (HCAs) for InfiniBand (IB). In the last few years, a new category of NIC has emerged known as Converged Network Adapter (CNA). The CNA is a NIC that supports FCoE in hardware.
A NIC is an essential component that connects a server to a network. At the core of the NIC is an ASIC that implements a Media Access Controller (MAC) and in many cases an integrated PHY. The PHY provides the media dependent interface physical layer while the MAC implements the Layer 2 function of the OSI model. The NIC also provides an interface to connect to the host hardware. In modern servers, this interface is based on Peripheral Component Interconnect Express (PCIe).

The PCIe link is provided by the Central Processing Unit (CPU) or chipset to feed into PCIe expansion slots or integrated devices. PCIe slots are used to expand the capabilities of the server via adapters such as NICs. Modern servers include embedded devices that are an integral subsystem on the main system motherboard. An Ethernet embedded device is called a LAN on Motherboard (LOM).

Figure 8 shows a conceptual depiction of the server networking stack. On the right, there are NIC hardware elements composed of the MAC and PHY. The MAC includes the PCIe block. The PHY consists of analog and digital components necessary for data communications over a noisy channel. The NIC connects to the host CPU via PCIe. Buffers and queues are used to transfer data between the NIC and system memory. The NIC software consists of a kernel mode driver, but other implementations such as user space drivers are available.
Figure 8. Simplified System Block Diagram

Figure 9 shows a physical depiction of the internal components of the host environment and how ports are represented at different layers of abstraction. The OS stack implements a network interface for each PCIe function. The PCIe subsystem performs two physical functions: one per physical port on Device 0. The LOM uses the NC-SI protocol and interface to provide shared LOM network access for the BMC. Then, the PHY implements digital and analog circuitry to allow transmission and reception of data.

Figure 9. The Network Edge

In modern server architectures, most of the PCIe links are provided by the CPU in a non-uniform memory access (NUMA) topology.
Central Processing Unit

The compute node or server has one or more CPUs. The CPU is the PCIe root complex and NUMA memory controller. The CPU feeds multiple PCIe lanes that can be used to connect embedded devices or PCIe expansion slots. It is recommended to put high performance NICs on PCIe links off the CPU.

Chipset

Additional PCIe lanes and I/O connectivity are provided by a companion ASIC. Low performance devices are connected to the host via components such as the Platform Controller Hub. Typical I/O includes SATA and USB ports. In some cases, the NIC MAC controller is integrated as well.

Memory

In modern servers, each CPU contains an integrated memory controller and attaches the controller directly to the system memory. In addition to the traditional Dynamic random-access memory (DRAM), modern systems offer flash memory to improve the system performance. The memory controller has been embedded in the CPU in the NUMA architecture. Each CPU has direct access to its local memory banks.

Peripheral Component Interconnect Express

PCle is a serial point-to-point link, packet-based protocol. Each link is made up of x number of lanes: x1, x2, x4, x8, and x16. Each lane uses two differential pairs (+/-): one to transmit and the other to receive traffic.

The transfer of packets occurs over a serial interface that is then byte striped across the lanes that form the link. Communications over the link occur between a requester and a completer.

Typically, the root complex in the CPU and an end node device at the opposite end communicate with each other. During link initialization, both devices establish the width and speed of the link before the link is operational.

As of the writing of this book, PCIe Gen 3 x8 and x16 are the links used for high performance peripheral I/O such as 10GE, 40GE, and 100GE. As indicated in Table 1, Gen 4 will be required to support speeds of 2x100 Gbps.
Table 1 describes the PCIe bandwidth requirements for various technologies for single port, dual port, and quad port PCIe adapters. It shows that starting with dual port 100GE, active/active PCIe Gen 4 x16 is required. For quad port NICs, Gen 4 x16 starting at 40GE is required. Quad port 100GE requires Gen 4 x32.

**Table 1. PCIe Bandwidth**

<table>
<thead>
<tr>
<th>Ports</th>
<th>GE</th>
<th>10GE FC8</th>
<th>25GE FC16</th>
<th>40GE FC32*</th>
<th>50GE FDR</th>
<th>100GE EDR STL1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Port</td>
<td>Gen1 x1</td>
<td>Gen2 x4</td>
<td>Gen2 x8</td>
<td>Gen3 x8</td>
<td>Gen3 x8</td>
<td>Gen3 x16</td>
</tr>
<tr>
<td>Dual Port</td>
<td>Gen1 x2</td>
<td>Gen2 x8</td>
<td>Gen3 x8</td>
<td>Gen3 x16</td>
<td>Gen3 x16</td>
<td>Gen4 x16</td>
</tr>
<tr>
<td>Quad Port</td>
<td>Gen1 x4</td>
<td>Gen3 x8</td>
<td>Gen3 x16</td>
<td>Gen4 x16</td>
<td>Gen4 x16</td>
<td>Gen4 x32</td>
</tr>
</tbody>
</table>

In addition, Table 1 specifies the type of PCIe interface required to support various technologies on dual port NICs and quad port NICs where all ports are active concurrently.

Table 2 stipulates the Ethernet bandwidths and the corresponding PCIe bandwidth coverage. It highlights the realized Ethernet bandwidth as a function of the theoretical Peripheral Component Interconnect (PCI) bandwidth.

For example, a quad port 40GE NIC running PCIe Gen 3 will be limited to 80% of its maximum theoretical network bandwidth. A dual port 100GE NIC running PCIe Gen 3 x16 will be limited to 64% of its maximum theoretical bandwidth.

**Table 2. PCIe Device Bandwidth Coverage**

<table>
<thead>
<tr>
<th>Ethernet</th>
<th>Dual Port</th>
<th>Quad Port</th>
<th>PCIe Gen3 x16 (128Gbps)</th>
<th>PCIe Gen3 x32 (256Gbps)</th>
<th>PCIe Gen4 x16 (512Gbps)</th>
<th>PCIe Gen4 x32 (512Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>2 Gbps</td>
<td>4 Gbps</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>10GE</td>
<td>20 Gbps</td>
<td>40 Gbps</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>25GE</td>
<td>50 Gbps</td>
<td>100 Gbps</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>40GE</td>
<td>80 Gbps</td>
<td>160 Gbps</td>
<td>DP-100% QP-80%</td>
<td>DP-100% QP-100%</td>
<td>DP-100% QP-100%</td>
<td>DP-100% QP-100%</td>
</tr>
<tr>
<td>100GE</td>
<td>200 Gbps</td>
<td>400 Gbps</td>
<td>DP-64% QP-32%</td>
<td>DP-100% QP-64%</td>
<td>DP-100% QP-100%</td>
<td>DP-100% QP-100%</td>
</tr>
<tr>
<td>200GE</td>
<td>400 Gbps</td>
<td>800 Gbps</td>
<td>DP-32% QP-16%</td>
<td>DP-64% QP-32%</td>
<td>DP-100% QP-100%</td>
<td>DP-100% QP-100%</td>
</tr>
<tr>
<td>400GE</td>
<td>800 Gbps</td>
<td>1600 Gbps</td>
<td>DP-16% QP-8%</td>
<td>DP-32% QP-16%</td>
<td>DP-32% QP-16%</td>
<td>DP-64% QP-32%</td>
</tr>
</tbody>
</table>
The specified maximum transfer rates are as follows:

- PCIe Gen 1 at 2.5 Gb/s
- PCIe Gen 2 at 5.0 Gb/s
- PCIe Gen 3 at 8.0 Gb/s
- PCIe Gen 4 at 16 Gb/s

These rates specify the raw bit transfer rate per lane in a single direction and not the rate at which data is transferred through the system. Effective data transfer rate or performance is lower due to encoding and framing overhead and other system design trade-offs such as maximum payload size.

Table 2 shows how the PCIe bandwidth affects the Ethernet link bandwidth for active/active dual port and quad port adapters. The table also shows where the PCIe bandwidth becomes a limitation for the various PCIe link technologies and Ethernet speeds. It shows that Ethernet speeds are outpacing the PCIe host interface.

**Baseboard Management Controller**

Servers include a service processor on the motherboard typically known as the Baseboard Management Controller (BMC). The service processor enables remote management of the server even when the OS is not running. A management console on the network can connect to the BMC via a shared LOM to save switch ports. A multiplexing circuit allows the LOM to carry OS traffic and BMC traffic. This communications path is based on a DMTF DSP0222 standard referred to as NC-SI over Reduced Media Independent Interface (RMII).

This sideband interface provides a pass-thru function where the LOM passes through traffic destined or sourced by the BMC. In addition, this sideband channel can be used by the BMC to manage the LOM by providing monitoring and inventory management operations.

Both OS to BMC and LAN to BMC traffic flows are supported. OS2BMC allows for local access communications between the host software and the BMC. For example, a browser can be pointed to the BMC web server IP address to launch the console locally. In addition, network communications between the BMC and a management console are supported via LAN2BMC.
Network Interface Card

The NIC is built from the building blocks shown in Figure 11.

Figure 10. NC-SI pass-thru

Figure 11. NIC Block Diagram
Table 3 describes the NIC components.

**Table 3. NIC BOM**

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Connector</td>
<td>RJ-45, SFP+, SFP28, QSFP+, QSFP28</td>
</tr>
<tr>
<td>Magnetic Transformer</td>
<td>Discrete, Integrated RJ-45-MAG</td>
</tr>
<tr>
<td>PHY</td>
<td>1000BASE-T, 10GBASE-T, 10GBASE-KR, 10GBASE-SR</td>
</tr>
<tr>
<td>MAC</td>
<td>GE, 10GE, 25GE, 40GE, 100GE</td>
</tr>
<tr>
<td>PCIe Serdes</td>
<td>Gen3 x8, Gen3 x16</td>
</tr>
<tr>
<td>NC-SI RGT</td>
<td>100Mbps</td>
</tr>
<tr>
<td>NVRAM</td>
<td>Flash, EEPROM</td>
</tr>
<tr>
<td>Voltage Regulators</td>
<td>Linear, Switching</td>
</tr>
<tr>
<td>Clock</td>
<td>Crystal/Oscillator</td>
</tr>
</tbody>
</table>
The NIC takes many forms including standard and proprietary designs to meet specific server platform requirements such as mechanical, electrical, and thermal considerations. In addition to the physical form factors, there are software implementations such as virtual adapters used with virtualized servers.

Server Interconnectivity

Historically, servers have provided a fixed LOM subsystem integrated directly on the main system planar as the primary network access. This provided a fixed physical layer interconnect, typically 1GbE and a fixed number of ports. This served customers well until the advent of high-speed 10GbE technologies, which provide significant benefits and physical-layer interface choices. As server designs became denser, it became inevitable that the built-in LOM will be replaced to offer more flexibility. Mainstream servers now require 10 Gbps interfaces with added demand for more flexibility from the LOM slot.

Server virtualization, network convergence, and space efficiency are the three primary factors driving the need for flexibility on the choice of LOM subsystem. In addition, each factor drives the need for higher bandwidth interconnects, such as 10GbE and 40GbE.

The server network interconnect landscape has changed significantly. The following points, along with Figure 12, help to explain these changes:

- The transition to 10GbE requires various physical interface choices. Both copper and optical options are needed, including 10GBASE-T and SFP+ optical and Direct Copper Attached. 10GBASE-T is the preferred physical interconnect option for installations requiring long cable runs (for example, >10 m). SFP+ and Direct Copper Attached is the current preference for top-of-rack 10GbE connectivity, but many installations are expected to shift to 10GBASE-T in the future. In addition, XAUI-to-KR transition is driven by blade infrastructure to provide 10GbE on all fabrics.

- Network convergence (iSCSI and FCoE) and virtualization is driving a richer feature set that offers choice of capabilities and performance.
The need for higher space efficiency means the network fabric taken by the LOM must be able to do more than just networking via Ethernet.

Traditional networking and storage server I/O has converged, broadening the supplier choice. The Network Daughter Card (NDC) provides link speed choice, network interface choice, and vendor choice.

**Figure 12. NIC Characteristics**

Hence, the server networking portfolio includes a rich set of devices that provide specific functionality on various form factors. The Ethernet Controller devices are implemented via various form factors as described in the following sections:

**Physical Classification**

The NIC comes in various mechanical forms and functions to meet the mechanical, power, and cooling design requirements of the server platform.
PCIe Adapter

The PCIe adapter is one of the form factors that houses the NIC and provides PCIe connectivity to the host. The PCI-SIG develops and maintains the PCIe standards. The standards define all the layers of the PCIe interface—physical, link (Data Link Layer Packet), and TLP. It also defines the mechanical and electrical specifications. The PCIe adapter plugs into standard PCIe slots. Current servers feature multiple PCIe standard expansion slots to support full-height and low-profile PCIe compliant adapters.

PCIe adapters come with corresponding brackets to secure the adapter in place. The PCIe Gen 3.0 standard specifies x1, x4, x8, and x16 links for adapters with corresponding wide connectors.

The four parts PCIe ID is assigned to the adapter based on its manufacturer. The device driver installation software uses the PCIe ID to select the right adapter supported by the driver.

The adapter can support an SMBus interface to communicate with the BMC for device inventory and monitoring operations.

The PCIe adapter is connected to a main planar over a PCIe link via pins on an edge connector. It typically connects to the network via 10GE SFP+ or 25GE SFP28 using Direct Attach Copper (DAC) or optical transceivers. In addition, it supports 1000BASE-T and 10GBASE-T over an RJ-45 connector.

LAN on Motherboard

A LOM is present in the baseline configuration of the server by default and is soldered down on the motherboard. It is the lowest cost form factor and provides shared LOM pass-thru traffic over NC-SI for OS2BMC and LAN2BMC.

On a LOM, the application-specific integrated circuit (ASIC) controller is integrated with the server main planar or motherboard and assumes the PCIe sub-system vendor ID and sub-system device ID of the main planar. The LOM controller four parts ID will change based on the system. The vendor ID and device ID reflect the ASIC semiconductor design.
LOM Riser Card

A LOM riser card is present in the base configuration of the server by default and cannot be changed. On a LOM Riser Card, the ASIC controller is designed on a separate card from the main planar or motherboard, but is specific to a given platform. It is designed on a separate card to meet the unique platform requirements when LOM or NDC is not feasible. The development and fulfillment model for a LOM Riser Card is the same as a LOM.

LOM risers are treated like a regular LOM. LOMs implement NC-SI for pass-thru communication between a management console on the network and a BMC. This is called shared LOM pass-thru. It saves switch ports when using a BMC dedicated port for network connectivity.

Network Daughter Card

The Network Daughter Card (NDC) is typically an OEM-specific form factor and provides PCIe connectivity to the host. It provides a flexible LOM since it can support various choices for the LOM including speeds, port count, and vendors. The NDCs implement NC-SI for pass-thru communication between a management console on the network and a BMC. The rack NDC is integrated with a main planar over a PCIe link via pins on a board-to-board connector. It typically connects to the network via 10 GE SFP+ or 25GE SFP28 using DAC or optical transceivers.

On an NDC, the ASIC controller is designed on an OEM custom card. There is always an NDC in the base configuration of the server. In this case, the LOM is flexible and there is choice at and after Point Of Sale.

![Figure 14. Rack NDC](image)

The Blade NDC is housed in a server blade chassis. It has KR or KR4 PHY that interfaces with an I/O Module—a switch or pass-thru. The other key interface is PCIe that is routed through a board-to-board connector between the NDC and the main planar in the blade server.
Mezzanine

A Blade Mezzanine is an expansion adapter for blade servers and is used as primary or secondary fabric. The blade mezzanine is treated as a PCIe expansion adapter with respect to the PCI ID strings, which are based on the mezzanine manufacturer. There is no NC-SI interface to the BMC either.

Blade mezzanines are proprietary PCAs elevated from the motherboard. They are used mostly on blade servers in lieu of PCIe adapters. The OCP mezzanine is another example. The Blade mezzanine is housed in a server blade chassis. It has typically KR or KR4 PHY that interfaces with an I/O Module—a switch or pass-thru. The other key interface is PCIe through a board-to-board connector to the main planar.
Virtual Classification

Virtual adapters are software constructs that are abstracted from the hardware. They come in various forms such as:

- vNIC
- VLAN
- VxLAN
- SR-IOV Virtual Function (Also known as macvtab or tab in Linux)

vNIC

Virtual NIC (vNIC) is implemented mostly by virtualization software. The vNIC is assigned to a VM. The VM requires the appropriate driver to be running and operational in the VM. The driver could be a regular driver that is not aware of the virtualization environment. On the other hand, the driver could be aware that it is running on a VM. This is known as para-virtualized driver. Typically, this driver enables stateless hardware offloads to improve performance. An example is VMXNET3 from VMware.

VLAN

Each tagged Virtual LAN (VLAN) is instantiated as a virtual adapter in the host OS networking stack and bound to TCP/IP as the upper layer protocol. A VLAN breaks up and isolates broadcast domains on a flat network. The NIC adds and removes the VLAN Tag prior to sending a frame up the stack.

VxLAN

Virtual Extensible LAN (VxLAN) encapsulates a Layer 2 frame for transmission over an L3 network via a tunneling mechanism. A VxLAN header is added that includes a 24-bit tag named Virtual Network ID (VNI). This tag is used to identify each tunnel. Legacy NICs cannot support the stateless offloads because of the VxLAN header encapsulation. Newer NICs were modified to add this support.
SR-IOV VFs

Single Root I/O Virtualization (SR-IOV) is a PCIe standard that defines a mechanism to break up a physical function into multiple virtual functions. The result is that one device can be shared across many virtual machines (VMs).

Virtual functions (VFs) provide independent DMA streams that can be assigned to a VM. A physical function includes the resource capabilities to create the VFs. A single Ethernet port appears as multiple devices showing in PCI configuration space as multiple functions. SR-IOV enabled hypervisors initialize and assign VFs to VMs. The VMs communicate directly with the NIC bypassing the hypervisor using the VFs. The outcome is higher throughput. The implementations of SR-IOV offload the data path to the NIC while maintaining the control path.

Functional Classification

The Ethernet controller devices are classified based on features and technologies such as:

**Network Interface Card**

It is an L2 device that implements protocols in the physical and link layer as well as stateless offloads. It is used as the interface between the server platform and the network.

**Converged Network Adapter**

This device implements L2 NIC functionality, but it adds L3 and L4 protocol offloads such as iSOE and FCoE. It is used as the interface between the server platform and the converged network.

**Host Bus Adapter**

It is used with FC as the interface between the server platform and the FC network.

**Host Channel Adapter**

It is used with IB as the interface between the server platform and the IB network.
Host Fabric Interface

It is used with Intel Omnipath Architecture as the interface between the server platform and the Omnipath network.

**Table 4. Device Functionality**

<table>
<thead>
<tr>
<th>Form Factors</th>
<th>Device Functional Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NIC</td>
</tr>
<tr>
<td>LOM</td>
<td>YES</td>
</tr>
<tr>
<td>LOM Riser Card</td>
<td>YES</td>
</tr>
<tr>
<td>NDC</td>
<td>YES</td>
</tr>
<tr>
<td>Adapter</td>
<td>YES</td>
</tr>
<tr>
<td>Mezzanine</td>
<td>YES</td>
</tr>
</tbody>
</table>
Data communications in the LAN involve the server OS and NIC, the switches, and a destination end node. The destination end node could be another server, a client, or a storage device.

**Host Perspective**

A NIC sends and receives frames between the end nodes over a network. At the source, application data is sent to the network stack in the host OS based on the destination socket, that is, IP address and TCP port number. The data is broken up into chunks of MSS length specified during TCP/IP connection setup. The network then fragments TCP packets into MTU size frames, if required. Normally, MSS is set to be lower than any MTU along the path to avoid fragmentation. The destination receives the packets in the controller on chip First In First Out (FIFO) buffer. From there, a PCIe DMA transaction copies the frame into host memory assigned to the NIC. A second memory copy places the data in the TCP buffer space. A final copy moves the data to the application memory space. Figure 17 illustrates the memory copies required to receive the frames. This figure shows one DMA copy with no CPU involvement and two CPU copies. Implementations will vary with the number of copies.
The host system sets up the memory space to store the header information to be consumed by each layer of the network stack. Memory is also allocated to hold the data payload as it is transferred to the application.

In addition, a buffer descriptor ring is implemented in the host memory to queue information based on the size and location of the frame memory store. The NIC reads the buffer descriptors to process frames for transmission or to receive incoming frames. The NIC sets buffer descriptors to inform the host where to get the received frames.

The received frame is buffered in the NIC FIFO and transmitted to the host memory via DMA. The NIC uses a descriptor ring buffer structure that specifies the amount and address of memory available for the NIC to transfer data to/from the host system via DMA. An interrupt is triggered to indicate that new data is available for processing.

An interrupt service routine runs on the host CPU assigned to handle the interrupt. This blocks any further interrupts from the NIC. The NIC driver executes a Deferred Procedure Call (DPC) to process the Rx descriptor queue.

The frame is copied to the application host memory after traversing the kernel stack via a CPU memory copy and CPU context switch from kernel to user space as shown in Figure 18. This figure shows one DMA copy with no CPU involvement and two CPU copies. Implementations will vary with the number of copies.

**Figure 17.** Packet Flow
As we will see further in this book, there are alternative methods to improve throughput, CPU utilization, and latency by bypassing the network stack, eliminating the buffer memory copies, and enabling Direct Data Placement (DDP) over a network, that is, RDMA. Other alternatives include moving network I/O transaction execution to user space to eliminate CPU context switching.

**Memory Buffers**

Host memory set up by the OS is used to store frame headers, frame data, and buffer descriptors. The host notifies the NIC of a buffer descriptor when it has data to send in memory. The NIC sends the frame via DMA.

The data is copied from the application memory to TCP/IP memory to NIC memory by using host CPU cycles. The NIC sends frames into its FIFO buffer via DMA.
Filters
The NIC controller implements L2 MAC address and VLAN filtering. Filters are employed to handle NPar and SR_IOV receive traffic.

To pass the L2 Ethernet MAC address filtering, the packet must be:
- Unicast – matches unicast MAC filter or Promiscuous mode is enabled
- Multicast – matches one multicast filter or Promiscuous multicast is enabled
- Broadcast – matches broadcast filter

A Receive packet that passes L2 layer MAC filtering successfully is subjected to VLAN header filtering. The VLAN tag must match the VLAN filter, or the packet will be discarded.

Queues
The transmit descriptor plane arbitrates between the different descriptor queues to determine which queue is serviced next.

On the receive side, the arbiter determines the order in which packets are written from the different packet buffers into the system memory.

The following filters/mechanisms determine the destination of a received packet:

Virtualization
In a virtual environment, DMA resources are shared between software elements (operating system and/or device driver). This is done by allocating receive descriptor queues to virtual partitions (VMs or VFs).

Data Center Bridging
Data Center Bridging (DCB) provides QoS through priority queues and priority flow control. Packets are classified into a Traffic Class (TC). Each TC is associated with a single unique packet buffer. Packets that reside in a specific packet buffer are then routed to an Rx queue based on their TC value.

Receive Side Scaling
Receive Side Scaling (RSS) distributes packet processing between several processor cores by assigning packets into different descriptor queues. RSS assigns an RSS index to each received packet. Packets are routed to an Rx queue based on its RSS index. An example is SR-IOV, which has queues per Virtual Function (VF). These are described in detail later in this book.
**Direct Memory Access**

Direct Memory Access (DMA) is used by the NIC controller ASIC to move received data from its FIFO buffer into the host main memory allocated to the NIC buffers and buffer descriptors without involving the CPU. The NIC becomes a PCIe Master that can transfer data to its allocated space in the host memory. The driver provides the necessary information such as memory address and size of the data. In most cases, the data payload is processed at various layers of the network stack in software. This requires that the data be moved to the applicable memory location. DMA is also used to move data to be transmitted by the NIC.

**Interrupts**

The NIC posts an interrupt whenever there is data to be processed. The host CPU is then interrupted to service the DMA transaction. Modern NICs use in-band messages via MSI/MSI-X over PCIe to signal an interrupt to the CPU. Interrupts are typically fired per MTU size frame. This would ensure the best latency, but at the cost of CPU utilization. NICs provide an interrupt moderation mode that coalesces multiple frames before firing an interrupt.

**Control Interfaces**

In addition to data interfaces such as PCIe and Ethernet, there are other auxiliary interfaces for the control plane, such as NC-SI and SMBus.

**NC-SI**

The Network Control Sideband Interface (NC-SI) is a Distributed Management Task Force (DMTF) standard that specifies a physical communication interface and a command-response protocol. The interface can be used to enable pass-thru traffic from a management console to the BMC over Ethernet such that a single Ethernet port can carry traffic for both the host OS and the BMC. NC-SI is typically implemented on server LOMs for pass-through traffic, LOM configuration, and status.

NC-SI has the capability to monitor and inventory the LOM hardware. In addition, a pass-thru channel has been defined between the host OS and the BMC.

The usable bandwidth for transmit and receive is up to 100 Mb/s in each direction.

NC-SI hardware arbitration (ARB) is used when there are multiple loads or packages known as Ethernet controllers sharing the same channel to the BMC. The ARB signals are routed between the controllers on the NDC. The RMII signals go to the BMC. However, the ARB signals stay on the NDC, which houses all of the packages.
**SMBus**

SMBus is an optional interface for monitoring the health and status of the NIC through an external BMC. In addition, SMBus is used by the BMC to maintain an inventory of all NICs installed in a system.

**I2C**

A serial management interface known as Inter-Integrated Circuit (I2C) is used for the control and management of external optical modules (XFP and SFP+). This interface enables the MAC controller and NIC software to monitor and control the state of the optical modules.

**Network Perspective**

Every Ethernet network interface that is not part of a bond team or bridge requires at least one unique Layer 2 address and at least one globally unique Layer 3 address to communicate across a network. Layer 2 is the Data Link Layer, and Layer 3 is the Network layer as defined in the OSI model. The Layer 2 address is assigned to the hardware and is often referred to as the MAC address or physical address. This address is pre-programmed at the factory and stored in NVRAM on a NIC or on the system motherboard for an embedded LAN interface.

The layer 3 addresses are referred to as the protocol or logical addresses assigned to the software stack. IP and Internetwork Packet Exchange (IPX) are examples of
Layer 3 protocols. In addition, Layer 4 (Transport Layer) uses port numbers for each network upper level protocol such as Telnet or File Transfer Protocol (FTP). These port numbers are used to differentiate traffic flows across applications. Layer 4 protocols such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) are most commonly used in today’s networks. The combination of the IP address and the TCP port number is called a socket.

Ethernet devices communicate with other Ethernet devices using the MAC address, not the IP address. However, most applications work with a host name that is translated to an IP address by a Naming Service such as Windows Internet Name Services (WINS) and Domain Name Services (DNS). Therefore, a method of identifying the MAC address assigned to the IP address is required. The Address Resolution Protocol (ARP) for an IPv4 network provides this mechanism.

For IPX, the MAC address is part of the network address and ARP is not required. However, for IPv4, ARP is implemented using an ARP Request and ARP Reply frame. ARP Requests are typically sent to a broadcast address while the ARP Reply is typically sent as unicast traffic. A unicast address corresponds to a single MAC address or a single IP address. A broadcast address is sent to all devices on a network.

Typical network enabled client/server applications use the Sockets API over well-known TCP or UDP port numbers and IP addresses to communicate over the network. The application opens a socket to initiate a TCP session to a destination IP address. However, text-based naming conventions are used to make it simpler for humans to remember the name of the destination.

A DNS server is used to translate the host name or FQDN to an IP address unless locally cached. At this point, the sender’s IP and MAC addresses and the destination’s IP address are known.

The destination’s MAC address needs to be determined as Ethernet requires physical L2 addresses for communications over the network. This mapping is done via ARP. The IP software calls on ARP to obtain the destination’s MAC address based on the destination’s IP address. ARP checks its cache first. If there is no entry, it sends an ARP request.

The IP address to be used as the target destination of the ARP request needs to be determined. These nodes will determine if the destination IP address is in the same subnet as the source by comparing their IP addresses using the subnet mask.

If the source and destination IP addresses are on the same subnet, then an ARP request is broadcasted with the destination IP address as the target to be resolved. The node with the matching IP address sends an ARP reply unicast with the corresponding MAC address.

On the other hand, if the source and destination nodes are on separate subnets, the ARP request is broadcasted with the gateway (router) IP address as the target to be resolved. The node with the matching IP address (the default gateway or default router) sends an ARP response unicast with the corresponding MAC address.
The Ethernet data frame is sent to the router MAC address as the destination. The router determines the best interface for reaching the destination IP address on another subnet. The router initiates an ARP request on the directly attached interface to determine the destination MAC address to be used in the data frame.

The NIC driver makes a function call to initiate data transmission once the TCP/IP stack has obtained the destination MAC address via an ARP resolution. The NIC generates an interrupt when it receives DMA data.

**Packet Buffers**

Ethernet controllers have packet buffering in the range of hundreds of KBs. Buffer space is implemented as FIFO with allocation for transmit and receive traffic. The size varies by implementation. Packet buffers are required to hold packets until the packet can be processed by the next stage. For receive path, packets may come at a faster rate than the host NIC can process via DMA.

Incoming packets are transferred from the packet buffers into data buffers in system memory. Data buffers are arranged around descriptor rings. Each descriptor queue is assigned dynamically to a packet buffer.

The packet plane, implements an arbiter at the output of the packet buffers, deciding which packet to transmit next.

**Spanning Tree Algorithm**

In Ethernet networks, only one active path can exist between any two bridges or switches. Multiple active paths between switches can cause loops in the network. When loops occur, some switches recognize stations on both sides of the switch. This situation causes the forwarding algorithm to malfunction, allowing duplicate frames to be forwarded.

Spanning tree algorithms provide path redundancy by defining a tree that spans all of the switches in an extended network and then forces certain redundant data paths into a standby (blocked) state. At regular intervals, the switches in the network send and receive spanning tree packets to identify the path. If one network segment becomes unreachable, or if spanning tree costs change, the spanning tree algorithm reconfigures the spanning tree topology and re-establishes the link by activating the standby path. Spanning tree operation is transparent to end stations, which do not detect whether they are connected to a single LAN segment or a switched LAN of multiple segments.

Spanning Tree Protocol (STP) is a Layer 2 protocol designed to run on bridges and switches. The specification for STP is defined in IEEE 802.1d. The main purpose of STP is to ensure that the user does not run into a loop situation when there are redundant paths in the network. STP detects/disables network loops and provides backup links between switches or bridges. It allows the device to interact with other STP compliant devices in the network to ensure that only one path exists between any two stations on the network.
Once a stable network topology has been established, all bridges listen for Hello Bridge Protocol Data Units (BPDUs) transmitted from the root bridge. If a bridge does not get a Hello BPDU after a predefined interval (Max Age), the bridge assumes that the link to the root bridge is down. This bridge then initiates negotiations with other bridges to reconfigure the network and reestablish a valid network topology. The process to create a new topology can take up to 50 seconds. During this time, end-to-end communications will be interrupted.

The use of Spanning Tree is not recommended for ports that are connected to end stations, because by definition, an end station does not create a loop within an Ethernet segment. Additionally, when a teamed adapter is connected to a port with Spanning Tree enabled, users may experience unexpected connectivity problems. For example, consider a teamed adapter that has a lost link on one of its physical adapters. If the physical adapter were to be reconnected (also known as fail-back), the intermediate driver would detect that the link has been reestablished and would begin to pass traffic through the port. Traffic would be lost if the port was temporarily blocked by the Spanning Tree Protocol.

To reduce the effect of topology changes on the network (for example, increasing flooding on switch ports), end nodes that are often powered on/off should use the Port Fast or Edge Port setting on the switch port they are attached to. Port Fast or Edge Port is a command that is applied to specific ports coming from link down to link up. This port will be put in the forwarding STP mode instead of going from listening to learning and then to forwarding. STP will still be running on the switch. The switch will not generate a Topology Change Notice when the port is going up or down.

**Figure 20.** STP
Performance

NIC performance is measured in terms of throughput, latency (sometimes includes jitter), and host CPU utilization.

- Latency is the time taken to receive the first bit of information. Memory copies, context switching, and the software network stack introduce latency. While latency is not a significant factor for most applications, it is critical to HPC, High Frequency Trading (HFT), and Virtual SAN (vSAN) workloads.

- Throughput is the time taken to receive all bits of information. Throughput is a function of many factors such as Ethernet link speed, PCIe link capacity, and internal data path design of the ASIC.

- Host CPU utilization can affect the performance of certain NICs. Most NICs require CPU cycles to copy data from the memory allocated to the NIC to the application buffer space. This often requires a CPU context switch from the kernel, where the network software stack resides, to a user space where most applications run.

Theoretical Performance

The Ethernet protocol defines 12 bytes inter-frame gap plus a MAC preamble of 8 bytes. The standard Ethernet frame consists of a MAC header 14 bytes long and a CRC field of 4 bytes. A minimum frame size of 64 bytes leaves 46 bytes for data payload. The maximum payload is 1500 bytes. However, when VLAN tags are used, an additional 4 bytes is added to the headers. Table 5 summarizes theoretical packet rates for various link speeds for the minimum and maximum MTUs.

Table 5. Packet Rates

<table>
<thead>
<tr>
<th>Link Speed (Gbps)</th>
<th>Packet Rates (pps)</th>
<th>Packet Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min MTU</td>
<td>Max MTU</td>
</tr>
<tr>
<td>10</td>
<td>14,880,952</td>
<td>812,744</td>
</tr>
<tr>
<td>25</td>
<td>37,202,381</td>
<td>2,031,860</td>
</tr>
<tr>
<td>40</td>
<td>59,523,810</td>
<td>3,250,975</td>
</tr>
<tr>
<td>100</td>
<td>148,809,524</td>
<td>8,127,438</td>
</tr>
</tbody>
</table>
PCI Express (PCIe) is the most prevalent interface used on standard x86 servers. PCIe is a high speed/full duplex serial interface that connects CPUs to peripheral devices such as the NIC. It is a point-to-point serial link connecting each device to the root complex, that is, host CPU and chipset. It is based on a layered approach that uses packetized structure for data delivery. These packets are called Transaction Layer Packets.

**PCIe Identification**

The NIC is programmed with a four part identifier. This identifier is formed of a PCIe vendor ID, device ID, subsystem vendor ID, and subsystem device ID. The IDs are used by the device driver to identify the right NIC that is supported by the driver. The driver has a list of PCIe device IDs that it supports. The driver will load only on a device that is included in the list. In some SR-IOV NICs, a different ID string for the VF is used to identify a VF driver when it is different from the PF driver.

**PCIe Link**

The NIC connects to the host system using a PCIe interface. The interface can be configured to operate in several link modes: x1, x2, x4, x8, and x16. A link between two devices is a collection of lanes. Therefore, an x2 link consists of two lanes. There are three generations of PCIe: 1.0, 2.0, and 3.0. A PCIe link consists of serial lanes. The bandwidth of each lane is specified based on the spec version 1.0, 2.0 or 3.0 to be 2.5, 5, or 8 Gbps respectively. For example, in PCIe 1.0, an 8b/10b technique is used, which means that for every 8 bits of data, 10 bits are transmitted on the wire. This translates to 20% overhead resulting in a data rate of 2 Gbps.

The physical link is managed by a local link traffic type known as Ordered Set. Ordered Sets do not go beyond a PCIe link. They are used during link training. In contrast, data in a PCIe packet is processed for delivery to the destination target.
Physical Layer

The physical layer performs the functions required to send and receive packets on the physical wire in the form of a stream of bits. On the receiver side, the serial stream is translated into a parallel format that is packaged into a frame. The opposite happens on the transmitter. A Phase Locked Loop (PLL) circuit is employed at the receiver side to extract the clock from the received data.

The transmitter implements the following functions: 8b/10b encoding, parallel to serial conversion, and scrambling to randomize the data stream to minimize Electromagnetic Interference (EMI). The receiver performs the inverse of these functions: decoding, serial to parallel conversion, and de-scrambling. In addition, it also implements a receiver PLL to recover the clock from data transmissions.

Initialization

The initialization phase auto-negotiates to the highest mutually supported lane count. The link can dynamically configure itself to use fewer lanes, providing failure tolerance in case bad or unreliable lanes are present.

Auto-negotiation establishes link parameters such as link width, lane frequency, and lane polarization. Software initiated re-training of PCIe link can be supported.

Enumeration

Discovery and identification of devices in a PCIe hierarchy is accomplished via a depth first software algorithm. This Bus enumeration is part of the system basic input/output system (BIOS).

Each link is given a Bus number, and a device on that link is Device 0 since there is only one device on a PCIe link. The Root Complex is Bus 0. Endpoint devices are often dual port or quad port up to 8 functions. Each port is identified via a function number.

Bus enumeration is performed by attempting to read the vendor ID register and device ID register for each combination of bus number and device number at the device’s function 0.

A device learns/stores the Bus/Device/Function (BDF) in the memory from a configuration Write packet. Devices with no BDF can only process configuration transactions.

Address Mapping

PCI devices are mapped into the system’s memory-mapped address space. The system’s BIOS programs the Base Address Registers (BARs) to inform the device of its address mapping by writing configuration commands to the PCI controller.
The NIC Ethernet controller implements various registers to hold or store commands, control, and status when power is on. These registers need to be mapped to the host system memory for interaction with the kernel driver software. This is called Memory Mapped IO (MMIO) space.

The Base Address Register (BAR) in the configuration header is used to specify the required space in memory. An expansion ROM uses a BAR to point to the address and size of memory. This is where the device initialization code resides. The system software copies this code into memory for execution by the host CPU.

**Transactions**

A requester sends a Transaction Lookaside Buffer (TLB) request packet to the completer to initiate communications. Information is exchanged by transferring one or more packets between the source (requester) and destination (completer).

Supported transaction categories include memory, I/O, configuration, and messages. These categories are further categorized into Posted and Non-posted. The key difference is that in non-posted transactions, the completer responds with a TLB completion message, acknowledging packet reception. In addition, non-posted reads include the data in the TLB completion. Conversely, non-posted writes include data in the TLB request.

Posted transactions were designed to optimize performance. The completer does not respond with a completion TLB message. It may or may not include data in the TLB request.

A requester manages many outstanding transactions by using a Tag. Both requester and completer include this Tag to match corresponding TLBs.

**Writes**

Memory Write TLP does not get an indication that the packet has reached its final destination, even less that it has been executed. Even if the Data Link Layer gets a positive acknowledgment, it only means that the packet made its way safely to the nearby switch. No end-to-end acknowledgment is ever made.

**Reads**

A Read occurs when the CPU wants to read data from a peripheral. Read operations are a bit trickier because there are two packets involved: One TLP from the CPU to the peripheral, asking the latter to perform a read operation, and one TLP going back with the data. In PCIe terms, we have a Requester (the CPU in our case) and a Completer (the peripheral).
Clocks

The device requires a 100 MHz differential reference clock, denoted PE_CLK_p and PE_CLK_n. This signal is typically generated on the system board and routed to the PCIe port. For add-in cards, the clock is furnished at the PCIe connector. PLLs are used to generate the necessary clocks.

Power

The PCIe standard defines the required operating power based on the type of adapter. The PCIe specifies the maximum power capacity as a function of card size and number of lanes per slot. There are 10 W, 25 W, and 75 W slots defined in the electromechanical specification for standard height and low profile cards.

Active State Power Management (ASPM) specifies ASPM L0s and ASPM L1 link state support. The NIC supports some operations when the server is operating in auxiliary power. In this mode, the NIC is in D3 stand-by power state and can support Wake on LAN and some management tasks over SMBus.

Flow Control

PCIe uses a credit-based flow control per Virtual Channel (VC) for posted, non-posted, and completion TLPs. It specifies frequency of flow control updates. The response to a target access does not depend on the status of a master request to the bus. If master requests are blocked (due to no credits), target completions can still proceed (if credits are available).

MSI/MSI-X

Message signaled interrupt (MSI) is the modern method to signal an interrupt to the CPU. MSI replaces the legacy hardware interrupts (designated as INTx) as the mechanism used by a device to interrupt the host CPU. The device that generated the interrupt sends a PME message with its identity upstream. The NIC specifies the number of vectors supported, target memory address, and data value. MSI is processed by the Interrupt controller in the Root complex. The CPU then proceeds to service the interrupt.

Power Management

NICs are designed to meet the requirements defined in PCI Power Bus Management Interface Specification 1.1, Advanced Configuration and Power Interface (ACPI) Specification 2.0, and PCI Express PM Link States (L0-L3). Device Power Management (D0-D3) is implemented to provide some functionality, but at a lower power.
The IEEE 802 is the LAN/MAN Standards Committee. There are various Working Groups within 802. This book focuses on the IEEE 802.3 Working Group. The 802.3 Working Group develops Ethernet standards. As of the writing of this book, the latest revision of the standard for Ethernet is IEEE Std 802.3-2015. There are currently 10 amendments to this standard. The IEEE Working Group is writing the following amendments:

1. IEEE P802.3bs 200 Gb/s and 400 Gb/s Ethernet Task Force
2. IEEE P802.3bt DTE Power via MDI over 4-Pair Task Force
3. IEEE P802.3ca 25 Gb/s, 50 Gb/s, and 100 Gb/s Ethernet Passive Optical Networks Task Force
4. IEEE P802.3cb 2.5 Gb/s and 5 Gb/s Backplane Task Force
5. IEEE P802.3cc 25 Gb/s Ethernet over Single-Mode Fiber Task Force
6. IEEE P802.3cd 50 Gb/s, 100 Gb/s, and 200 Gb/s Ethernet Task Force
7. IEEE P802.3-2015/Cor 1 (IEEE P802.3ce) Multilane timestamping
8. IEEE P802.3.2 (IEEE P802.3cf) YANG Data Model Definitions Task Force
9. IEEE P802.3cg 10 Mb/s Single Twisted Pair Ethernet Task Force
10. IEEE P802.3ch Multi-Gig Automotive Ethernet PHY Task Force

As of the writing of this book, the following are the link speeds supported by Server Ethernet IEEE compliant NICs along with its supported media types:

- 1GE (RJ-45/1000BASE-T)
- 10GE (KR, SFP+, RJ-45/10GBASE-T)
- 25GE (KR, SFP28, MMF Optics/SR, CR, DAC)
40GE (KR4, QSFP+)
100GE (KR4, KP4, QSFP28)

**NOTE:** All 802 specifications are available free of charge from the IEEE at [http://standards.ieee.org/about/get/](http://standards.ieee.org/about/get/), six months after publication.

Ethernet networks continue to evolve their capabilities. Link speeds continue to increase via advances in physical layer DSP to maximize the bandwidth capacity of various transmission channels. In addition to 10GE controllers, there are 40GE controllers based on 4x10GE lanes and 100GE controllers based on 4x25GE. In addition, 25GE has become the newest IEEE Std 802.3by-2016 standard. It is expected that 25GE will grow past 40GE. Unlike 40GE, which requires four lanes, 25GE requires a single lane. This provides greater bandwidth and higher density switch designs.

### Table 6. NIC Speeds

<table>
<thead>
<tr>
<th>EN Speeds</th>
<th>Throughput (Gbps)</th>
<th>Line Rate (Baud)</th>
<th>Encoding</th>
<th>PCIe Dual Port Card</th>
</tr>
</thead>
<tbody>
<tr>
<td>1G</td>
<td>1</td>
<td>125M</td>
<td>8b/10b</td>
<td>PCIe 1.0 x2</td>
</tr>
<tr>
<td>10G</td>
<td>10</td>
<td>800M</td>
<td>64b/66b</td>
<td>PCIe 2.0 x8</td>
</tr>
<tr>
<td>25G</td>
<td>25</td>
<td>25.78125G</td>
<td>64b/66b</td>
<td>PCIe 3.0 x8</td>
</tr>
<tr>
<td>40G</td>
<td>40</td>
<td>10.3125G</td>
<td>64b/66b</td>
<td>PCIe 3.0 x16</td>
</tr>
<tr>
<td>100G</td>
<td>100</td>
<td>25.78125G</td>
<td>64b/66b</td>
<td>PCIe 4.0 x16</td>
</tr>
</tbody>
</table>

**GE**

The GE IEEE Std 802.3-2015 specifies both copper and fiber media PHYs as follows:

- 100 m on CAT-5 via copper PHYs
- 550 m on MMF (SX) via optical PHYs
- 5 km on SMF (LX) via optical PHYs

Multiple speeds are supported via Auto-negotiation. Implementations from various vendors include 100 Mb/s and 1000 Mb/s via copper PHY. Others support 10 Mb/s, 100 Mb/s, and 1000 Mb/s on the same PHY. Negotiating to a lower speed allows for network communications in lower power scenarios like Wake on LAN.

1000BASE-T included both full and half duplex in the standard. However, implementations only delivered full duplex.
The first 10GE standard included only fiber links using Multi-mode Fiber (MMF) and Single-mode Fiber (SMF). No auto-negotiation was defined. The links ran only at 10 Gbps.

Carrier Sense Multiple Access/Collision Detection (CSMA/CD) half-duplex operation was not supported.

Figure 21 shows the various layers for MAC and PHY. The layers are further defined in Table 7.
Table 7. Ethernet Layers

<table>
<thead>
<tr>
<th>Layer</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCS</td>
<td>Physical Coding Sublayer</td>
<td>• Encoding/decoding of XGMII parallel lanes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Link initialization and status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Transfers data to/from the PMA</td>
</tr>
<tr>
<td>PMA</td>
<td>Physical Medium Attachment</td>
<td>• TX Serialization/RX Deserialization of PMD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Clock recovery on receive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mapping of TX/RX bits</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Loopback Mode</td>
</tr>
<tr>
<td>PMD</td>
<td>Physical Medium Dependent</td>
<td>• Transceiver for transmission and reception over physical medium</td>
</tr>
<tr>
<td>MDI</td>
<td>Medium Dependent Interface</td>
<td>• Ethernet over twisted pair</td>
</tr>
<tr>
<td>RS</td>
<td>Reconciliation Sublayer</td>
<td>• Adapts bit serial protocols of MAC to parallel encoding of PHY</td>
</tr>
<tr>
<td>XGMII</td>
<td>10Gb Media Independent Interface</td>
<td>• Data is synchronous to clock reference</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 32-bit wide independent transmit and receive data paths</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Full-duplex only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Each direction serviced by independent data, control and clock signals</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Data: TXD&lt;31:0&gt;, RXD&lt;31:0&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Control: TXD&lt;3:0&gt;, RXD&lt;3:0&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Clock: TX_CLK, RX_CLK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Supports only 10Gb/s MAC data rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Maximum length: 7 cm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number of signals: 74</td>
</tr>
</tbody>
</table>

The 10GE Physical Medium Dependent (PMD) includes support for short range (SR), long range (LR), and extended range (ER) transceivers over MMF and SMF optical media.

Table 8. 10GE PMD

<table>
<thead>
<tr>
<th>Type</th>
<th>Optical PMD</th>
<th>Fiber Media (nm)</th>
<th>Operating Range (m)</th>
<th>Speed (Gb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN Serial</td>
<td>10GBASE-SR</td>
<td>850</td>
<td>2-300 (50um)</td>
<td>10.3125</td>
</tr>
<tr>
<td>10GBASE-LR</td>
<td>1310</td>
<td>2-10,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10GBASE-ER</td>
<td>1550</td>
<td>2-40,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN WDM</td>
<td>10GBASE-LX4</td>
<td>1300</td>
<td>2-300</td>
<td>12.5</td>
</tr>
<tr>
<td>10GBASE-X</td>
<td>1310</td>
<td>2-10,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAN Serial</td>
<td>10GBASE-SW</td>
<td>850</td>
<td>2-300 (50um)</td>
<td>9.95328</td>
</tr>
<tr>
<td>10GBASE-LW</td>
<td>1310</td>
<td>2-10,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10GBASE-EW</td>
<td>1550</td>
<td>2-40,000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9 summarizes the supported fiber media including multi-mode fiber and single-mode fiber. SMF supports longer fiber cables than MMF.
The server NIC implements support for the interfaces in the LAN and Data Center zone as shown in Figure 22. The most common interface is still 1000BASE-T, but 10GE is rapidly becoming the default interface on mainstream servers in both SFP+ and 10GBASE-T form factor. 10GBASE-KR is the most utilized interface on modular servers traversing mid-plane and connectors inside a chassis to connect the server to the I/O module. 10GBASE-T is supported up to 100 m over CAT6a while the twinax DAC reaches up to 15 m theoretically, although in practice most DAC cables are 3 m, 5 m, or 7 m long. For longer reach, the standards define support for optical interconnects.

**Figure 22.** 10GE Applications

**Table 9.** 10GE Fiber Media

<table>
<thead>
<tr>
<th>PHY</th>
<th>50um MMF</th>
<th>62.5um MMF</th>
<th>10um SMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>10GBASE-SR</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>10GBASE-LR</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>10GBASE-ER</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>10GBASE-SW</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>10GBASE-LW</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>10GBASE-EW</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>10GBASE-LX4</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
10GBase-T PHY

10GE over a UTP CAT6a cable up to 100 m proved to be a significant jump in the physical layer design. Therefore, CAT6a was specified up to 625 MHz. CAT6a increased the number of twists for the wires at different rates per pair.

Figure 23 illustrates a conceptual block diagram of the 10GBASE-T PHY showing the key functional blocks.

![Figure 23. PHY Layer Blocks](image)

In addition, the IEEE Std 802.3-2015 specifies the following techniques:

- The Self-synchronizing scrambler is needed to control EMI radiation by randomizing the clock transitions.

- A 2-dimensional 128 DSQ (Double Square QAM) constellation conveys 7 bits per symbol modulation technique.

- Low Density Parity Check (LDPC) is a block error correcting code with block size of 2048 bits, with 325 check bits: a (2048, 1723) code.

- Tomlinson-Harashima Precoding (THP) is a scheme in which the equalizer for the channel is placed in the transmitter.

25GE

There are two versions of 25GE NICs—25G Ethernet Consortium pre-standard and IEEE Std 802.3by-2016.

The 25G Ethernet consortium defined the logical layers (PCS/PMA) of 25G/50G PHYs. It leveraged the 100G IEEE specs (IEEE 802.3bj, IEEE 802.3bm). It includes the required auto-negotiation to determine the link speed and Forward Error Correction (FEC). The consortium version defined both 25GE and 50GE (2x25 Gbps) PHYs over copper twinax cables only.
The IEEE version only supports 25 Gbps including DAC twinax cables and optical cables. The IEEE Std 802.3by-2016 defines three data-path modes for error correction on 25GE: No-FEC, BASE-R FEC (Firecode), and RS-FEC (Reed Solomon).

The IEEE Std 802.3by-2016 and the 25G Ethernet consortium are similar with two exceptions:

- **Auto-negotiation**
  - IEEE Std 802.3by-2016 uses the technology bits (25GBASE-CR, 25GBASE-CR-S) and the FEC base page
  - 25G consortium uses an Organizational Unique Identifier (OUI) next page using the 25G consortium OUI
  - RS-FEC/PCS AM markers insertion arch model (semantics, does not impact the data)

- **IEEE 25G and 25G consortium can be supported on the same device, including auto-negotiation as follows:**
  - A port that supports both IEEE and consortium will advertise both the IEEE technology bits and the consortium OUI page
  - If the link ends support both IEEE 25G and consortium 25G, IEEE takes priority over the 25G consortium

Early 25GE NICs only supported one link speed—25 Gbps. Some lacked Reed Solomon FEC. Newer generations were fully IEEE Std 802.3by-2016 compliant.

RS-FEC is needed to support 5 m DAC cables and 25G-SR optical specification since reuse of 100G-SR4 single-lane specs is sought. The Firecode FEC is expected to be of lower latency than RS-FEC for 25GE and useful for 1 m and 3 m DAC cables.

A no FEC option is potentially applicable to 1 m cable and backplane applications.
IEEE 802p.3bq 25G/40GBASE-T was ratified in June 2016. Both 25GBASE-T and 40GBASE-T have the same targeted cabling – 30 meter, 2 connector Category 8 cabling. 25GBASE-T is running at 2.0 Gbaud and specifies Category 8 characteristics out to a frequency of at least 1250 MHz, while 40GBASE-T specifies to 2000 MHz.

The Task Force rejected an attempt to add Category 7a as a specified medium for 25GBASE-T. RJ-45 remains the only Medium Dependent Interface (MDI) in the specification for 25GBASE-T as well as for 40GBASE-T.

### 25G/40GBASE-T

Higher speeds are supported via QSFP connectors on DAC twinax. Server NICs implement 40GE via 4x10GE MLD and 100GE via 4x25GE MLD as specified in IEEE Std 802.3-2015.

100GE (4x25) optical support via SMF up to 40 Km is defined as shown in Figure 25. In addition, DAC twinax support is defined up to 7 m cables.

40GE (4x10) is defined to run over optical cable, DAC twinax cable, and 1 m over a chassis backplane.
IEEE P802.3cd will define 50 Gb/s, 100 Gb/s, and 200 Gb/s using PAM4 50 Gb/s lanes. Current implementations of 50 Gb/s and 100 Gb/s are based on 25 Gb/s SerDes specifications.
Link settings are parameters configured on both ends of the link. Typically, they must match to ensure correct behavior of the link.

**Duplex**

The original Ethernet was only half duplex based on a shared link using CSMA/CD. Modern switched Ethernet technologies are full duplex. This means that there are circuits capable of transmitting and receiving bits at the same time.

**Wake on LAN**

A network wake-up event is a request from an external hardware or software to place the system into a fully powered state (S0, working) from a lower power state.

The Network Device Class Power Management Reference Specification defines three methods of generating wake-up events:

- Method #1: Detection of a change in the network link state
- Method #2: Receipt of a network wake-up frame
- Method #3: Receipt of a Magic Packet

**Auto-negotiation**

The Auto-negotiation standard allows devices based on several Ethernet standards, from 10BaseT to 10GBaseT, to coexist in the network by mitigating the risks of network disruption arising from incompatible technologies. This capability helps in ensuring a smooth migration path from Ethernet to Fast Ethernet, Gigabit Ethernet, and 10GBase-T. This section provides an in-depth explanation of auto-negotiation and its operation and also discusses special cases that may be encountered.
Many technologies, such as 10BaseT, 100BaseTX, 1000BaseT, and 10GBase-T, use the same RJ-45 connector, thus increasing the risk of connecting electrically incompatible components together and causing network disruption. In addition, with the advent of 10 Gigabit Ethernet over copper, multiple-speed devices now support 10 Mbps, 100 Mbps, 1000 Mbps, and 10 Gbps operations. The Institute of Electrical and Electronics Engineers (IEEE) developed a method known as auto-negotiation to eliminate the possibility of dissimilar technologies interfering with each other.

Gigabit transceivers at the PHY of the OSI model use auto-negotiation to advertise the following modes of operation: 1000BaseT in full or half duplex, 100BaseTX in full or half duplex, and 10BaseT in full or half duplex. Although auto-negotiation can be disabled for 100BaseTX or 10BaseT connectivity, it is always required for normal 1000BaseT operation.

Auto-negotiation enables an easy upgrade path to 10 Gigabit speeds by future proofing the server network connectivity with a three-speed NIC or LOM. A server connected to a Fast Ethernet switch or hub can easily be upgraded to Gigabit Ethernet by connecting the NIC to a Gigabit Ethernet switch. If both the NIC and the switch are set to auto-negotiate, the interface will be automatically configured to run at 1000 Mbps.

The auto-negotiation algorithm (known as NWay) allows two devices at either end of a 10 Mbps, 100 Mbps, 1000 Mbps, or 10 Gbps link to advertise and negotiate the link operational mode, such as the speed of the link and the duplex configuration, to the highest common denominator.

In addition, auto-negotiation determines the master-slave mode between the PHYs at the ends of the link. This mode is necessary to establish the source of the timing control of each PHY. Auto-negotiation is an enhancement of the 10BaseT link integrity test (LIT) signaling method and provides backward compatibility with link integrity.

Auto-negotiation is defined in Clause 28 of the 1998 edition of the IEEE Std 802.3. The current IEEE 802.3 standard includes Clause 37 and Clause 73 in addition to Clause 28. Clause 28 defines auto-negotiation for twisted pair links. Clause 37 describes the 1000BASE-X Auto-Negotiation (AN) function. AN for Backplane Ethernet is specified in Clause 73. Auto-negotiation provides the following benefits:

- Provides easy, plug-and-play upgrades from 10 Mbps, 100 Mbps, 1000 Mbps and 10 Gbps as the network infrastructure is upgraded
- Prevents network disruptions when connecting mixed technologies such as 10BaseT, 100BaseTX, and 1000BaseT
- Accommodates future PHY (transceiver) solutions
- Allows manual override of auto-negotiation
- Supports backward compatibility with 10BaseT
- Provides a parallel detection function to recognize 10BaseT, 100BaseTX, and 100BaseT4 non-NWay devices
In addition, the 1999 standard for Gigabit over copper cabling, IEEE Std 802.3ab, added the following enhancements to the auto-negotiation standard:

- Mandatory auto-negotiation for 1000BaseT
- Master and slave modes configuration for the PHY

The auto-negotiation specification includes reception, arbitration, and transmission of normal link pulses (NLPs). It also defines a LIT function for backward compatibility with 10BaseT devices. All these functions are implemented as part of the physical layer transceiver as shown in Figure 26. The exchange of link information occurs between the PHY and the MDI or RJ-45 connector.

![Figure 26. Layer Model (Redrawn from the IEEE Std 802.3, 1998 Edition)](image)

Gigabit Ethernet defines auto-negotiation as a functional block part of the physical coding sublayer (PCS) function, while in 100BaseT, it is defined as a separate sublayer in the PHY. All auto-negotiation functions are implemented as part of the transceiver integrated circuit, which is part of a NIC or integrated on the motherboard of a computer.

**10BaseT Link Test Pulses**

The 10BaseT standard includes a link test mechanism to ensure network integrity. In the absence of network traffic, a 100 nanosecond (ns) heartbeat unipolar (positive only) pulse is sent every 16 milliseconds (ms) within a range of +/- 8 ms. These pulses are called Normal Link Pulses (NLP). The link test pulse is sent by the transmitters of all 10BaseT media attachment units (MAUs) between the data terminal equipment (DTE) and the repeater.

A link fail condition is entered if the receiver does not receive a packet or a link test pulse within 50-150 ms. The link fail condition disables the data transmit, data receive, and loopback functions. The link test pulses continue to be transmitted and received during the link failure. The link is reestablished when two consecutive link test pulses or a single data packet has been received.
100BaseT/1000BaseT Fast Link Pulses

The link information is encoded in a special pulse train known as the Fast Link Pulse (FLP) burst. The FLP builds on the LIT pulse used by 10BaseT devices as a heartbeat pulse to the link partner at the opposite end of the link. The LIT was redefined as the normal link pulse (NLP). As shown in Figure 27, the NLP is the 10BaseT LIT pulse, and the FLP is a group of NLPs. Each pulse is 100 ns in width.

![Figure 27. FLP and NLP Comparison](image)

Auto-negotiation replaces the single 10BaseT link pulse with the FLP burst. Auto-negotiation stops the transmission of FLP bursts once the link configuration is established. The FLP burst looks the same as a single link test pulse from the perspective of 10BaseT devices. Consequently, a device that uses NWay must recognize the NLP sequence from a 10BaseT link partner, cease transmission of FLP bursts, and enable the 10BaseT physical medium attachment (PMA). Auto-negotiation does not generate NLP sequences—it only recognizes NLPs. Instead, auto-negotiation passes control to the 10BaseT PMA to generate NLPs.
FLP Bursts

Each FLP burst consists of 33 pulse positions that provide clock and data information. The 17 odd-numbered pulses are designated as clock pulses, while the 16 even-numbered pulse positions represent data information. A logic one is represented by the presence of a pulse, while the absence of a pulse is represented by a logic zero. Figure 28 shows the timing characteristics of the clock and data pulses.

![Figure 28. FLP Burst Timing](image)

FLP Burst Encoding

The data pulses in the FLP burst encode a 16-bit link code word (LCW). A device capable of auto-negotiation transmits and receives the FLP. The receiver must identify three identical LCWs before the information is authenticated and used in the arbitration process. The devices decode the base LCW and select capabilities of the highest common denominator supported by both devices. Once the LCWs are properly received, each device transmits an FLP burst with an acknowledge bit. At this point, both devices enable the highest common mode negotiated.

The clock pulses are used for timing and recovery of the data pulses. The 17 clock pulses are always present in the FLP burst. The first pulse on the wire is a clock pulse. The 16 data pulses may or may not be present. If the data pulse is present, it represents a value of one in the LCW for that position. The lack of a data pulse indicates a zero in the LCW for that position, as shown in Figure 29.

![Figure 29. FLP Burst Encoding](image)
Base Link Code Word

The base Link Code Word (LCW) is transmitted within an FLP burst after power-on, reset, or renegotiation of the link speed and duplex setting. The DTE and its link partner communicate their capabilities by exchanging LCWs. Figure 30 defines the bit positions of the base LCW. These bit positions map directly to the data pulses in the FLP burst-bits D0 through D15.

![Base Link Code Word Bit Positions](image)

Technology Ability Field

The technology ability field (TAF), which is encoded in bits D5 through D12 of the FLP burst, is shown for the IEEE Std 802.3-2015 Base Page as defined in the Selector field (00001) for IEEE 802.3 Ethernet. The order of the bits within the TAF does not correspond to the relative priority of the devices during the arbitration process. Each device capable of auto-negotiation maintains a prioritization table used to determine the highest common denominator ability. These priorities were updated to include Gigabit Ethernet over copper, as listed in Table 10.
Table 10. Priority Resolution

<table>
<thead>
<tr>
<th>Priority</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (highest)</td>
<td>1000BaseT – Full duplex</td>
</tr>
<tr>
<td>2</td>
<td>1000BaseT – Half duplex</td>
</tr>
<tr>
<td>3</td>
<td>100BaseT2 – Full duplex</td>
</tr>
<tr>
<td>4</td>
<td>100BaseTX – Full duplex</td>
</tr>
<tr>
<td>5</td>
<td>100BaseT2 – Half duplex</td>
</tr>
<tr>
<td>6</td>
<td>100BaseT4</td>
</tr>
<tr>
<td>7</td>
<td>100BaseTX – Half duplex</td>
</tr>
<tr>
<td>8</td>
<td>10BaseT – Full duplex</td>
</tr>
<tr>
<td>9 (lowest)</td>
<td>10BaseT – Half duplex</td>
</tr>
</tbody>
</table>

**Gigabit Auto-negotiation**

1000BaseT devices use auto-negotiation to set up the link configuration by advertising the PHY capabilities, including speed, duplex, and master-slave mode. Gigabit Ethernet over copper relies on the exchange of a Next Page LCW that describes the Gigabit extended capabilities.

A 1000BaseT PHY can operate as a master or slave. A prioritization scheme determines which device will be the master and which the slave. The IEEE supplement to IEEE STD 802.3ab-1999 Edition defines a resolution function to handle any conflicts. Multiport devices have higher priority to become master than single port devices. If both devices are multiport devices, the one with higher seed bits becomes the master.

**Parallel Detection**

A device determines if a link partner can use auto-negotiation by detecting the FLP burst. However, some devices may not have implemented the auto-negotiation function. For devices that support 100BaseTX, 100BaseT4, or 10BaseT, a parallel detection function allows detection of the link speed by detecting the NLP or FLP.
Duplex Mismatching

A link is degraded when the duplex settings (half duplex and full duplex) of a device and its link partner, such as a server NIC and a switch, do not match. When both devices send frames simultaneously on the link in a mismatched duplex configuration, the following conditions occur:

- The half-duplex link detects a collision, which corrupts its outgoing frame and discards the incoming frame. The half-duplex link will attempt to retransmit the frame.
- The full-duplex link will not resend its frame. It determines that the incoming frame is bad and flags cyclic redundancy check (CRC) errors.
- Applications will time out and retransmit continuously, causing a very slow connection.

Enabling Migration to Gigabit Speeds

The IEEE standard for auto-negotiation ensures easy migration from 10 Mbps to 100 Mbps, 1000 Mbps, 10 Gbps and higher speeds. A 10/100 BASE-T NIC installed on a server connected to a 10/100 switch port set for auto-negotiation can be upgraded to a 10/100/1000 BASE-T connection by simply replacing the NIC in the server. The new NIC will auto-negotiate to 100 Mbps full duplex automatically. The same is true with a LAN-on-motherboard (LOM) interface. A new server with a 10/100/1000 BASE-T LOM interface can be substituted with no configuration changes in a switch or cable plant.

When a 10/100 Ethernet switch is upgraded to a Gigabit over copper switch, the NIC in the server will negotiate to operate at 1000 Mbps automatically, without stopping or rebooting the server. This is true for 10GBase-T and future standards such as 25GBase-T.

Maximum Transmission Unit

The Ethernet standard defines Maximum Transmission Unit (MTU) as the maximum frame size that can be transmitted on an Ethernet network. The IEEE specifies 1500 bytes as the standard size MTU. However, other MTU sizes have been implemented to address the need for larger MTU size. These can be 9 KB jumbo frames or 2.5 KB for FCoE. The key is that the MTU size must be set to the same value at every node in the network traverse from source to destination.
Flow Control

This is a link layer congestion control mechanism to reduce frame discards between two ends of an Ethernet full duplex link. Pause frames are sent and/or received on an Ethernet Port when a certain buffer threshold is reached. The transmitter will cease to transmit until a Pause=0 is received.

Data Center Bridging

DCB introduces support for multiple traffic classes (TCs) assigning different priorities and bandwidth per TC. Link level Flow Control (PAUSE) IEEE 802.3x stops all the traffic classes. Priority-based Flow Control (PFC) allows more granular flow control on the Ethernet link in a DCB environment as opposed to the STD PAUSE mechanism.

PFC is implemented to prevent receive packet buffers overflow. Receive packet buffers overflow results in the dropping of received packets for a specific TC. PFC is designed to indicate to the transmitter to stop transmitting packets for that TC until the XOFF timer expires or an XON message is received for the stopped TC.

DCB is a set of standards developed via the IEEE to enable a near lossless behavior that supports new protocols such as Fibre Channel over Ethernet (FCoE) and RDMA over Converged Ethernet (RoCE).

In short, DCB enables Ethernet fabrics to support lossless flows required during congestion. There are two popular versions: a pre-standard supported by a consortium of companies and an IEEE standard. Most new implementations are using the IEEE version.

The pre-standard version, CEEv0, was implemented prior to the IEEE standards. These two versions are incompatible with each other. Current implementations will likely support both versions. They will try running the IEEE version first, followed by CEEv0.

DCB includes the following protocols:

- Priority Flow Control
- Enhanced Transmission Selection
- Congestion Notification
- Data Center Bridging Capability Exchange Protocol

End-to-end Congestion Notification has not been adopted to any significant extent and there is no congestion avoidance like what TCP provides. DCB only provides notification once congestion is detected.
DCB development is focused on FCoE, but it can benefit iSCSI by improving handling of steady state and transient network congestion conditions.

Table 11. DCB Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Standard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority Flow Control (PFC)</td>
<td>IEEE 802.1Qbb</td>
<td>• Enables multiple traffic types to share a common ethernet physical link without interfering with each other</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Allows link flow control to be performed on a per-priority basis</td>
</tr>
<tr>
<td>Enhanced Transmission Selection (ETS)</td>
<td>IEEE 802.1Qaz</td>
<td>• Enables transmission bandwidth management/rate limiting per traffic class</td>
</tr>
<tr>
<td>Congestion Notification (CN)</td>
<td>IEEE 802.1Qau</td>
<td>• End-to-end congestion management of long-lived data flows</td>
</tr>
<tr>
<td>Data Center Bridging Capability Exchange Protocol (DCBCX)</td>
<td>IEEE 802.1Qaz</td>
<td>• LLDP-based protocol to exchange link configuration parameters for PFC, ETS, etc.</td>
</tr>
</tbody>
</table>
Figure 31 shows the position of MAC and PHY functional blocks on the NIC. Typically, the MAC and PHY are integrated on a single die and ASIC package.

![NIC Block Diagram](image)

**Media Access Controller**

The MAC implements the Layer 2 logic blocks, as defined by the IEEE 802.3 standards. It has Rx and Tx DMA Engines. A PCIe SerDes block provides host connectivity. The MAC includes an interface to the PHY. FIFO provides for a packet buffer.
Interrupt Moderation

Interrupt Moderation is a technique to reduce the number of interrupts used to process frames. Typically, the host CPU must be interrupted to execute software to handle the processing of frames. Frames can be received one at a time, thus creating a large number of interrupts that the CPU must process but offering the lowest latency. On the other hand, if one wants to optimize for CPU utilization, then interrupt moderation can be used. In this case, frames are pooled together allowing a single interrupt to process multiple frames.

Wake on LAN

Wake on LAN is a feature that allows a system to be awakened from a sleep state by the arrival of a specific packet over the Ethernet interface. Since a virtual adapter is implemented as a software-only device, it lacks the hardware features to implement Wake on LAN. However, the physical adapters support this property even when the adapter is part of a team. The PCIe block in the MAC asserts the WAKE signal to indicate that the system should turn on the main power and the OS should become active.

Direct Memory Access

When the NIC is initialized, it requests for system resources such as MMIO and MSI vector(s). The host system allocates DRAM system memory to process the frames as they move through the network software stack to the destination, that is, application. Prior to DMA, the CPU would get involved in moving data from the NIC to the host.
system memory buffers allocated to NIC. The DMA controller performs Direct Memory Access without involving the CPU in data movement.

**Precision Timing Protocol**

Some applications could benefit from having a synchronized time of the day in compute, storage, and networking elements, such as HFT. The NIC can support Precision Time Protocol (PTP) in combination with the host software stack.

The operation of a PTP enabled network is divided into two stages: initialization and time synchronization.

At the initialization stage, every master-enabled node starts by sending sync packets that include its clock parameters. The nodes compare the received value to its current value. Eventually, the best master clock is chosen.

The time synchronization stage is when the master and slave nodes on the network, servers and switches for example, exchange time stamped messages for time offset calculation.

Typically, the NIC hardware implements the following functions:

- Determines the packets that require time stamping
- Time stamps the packets on both Rx and Tx path
- Stores the time stamp value for software
- Keeps the system time in hardware

On the other hand, the software in the OS stack performs the following functions:

- Defines the node state (master or slave) and selection of the master clock if in slave state
- Generates and consumes PTP packets
- Calculates the time offset and adjusts the system time

**Physical Layer**

The physical transceiver implements the media dependent interface compliant with IEEE Std 802.3-2015. It converts digital data into analog waveforms for transmission over a communication channel such as copper cable, optical cable, or backplanes. It also converts received signals from analog to digital. Twisted pair copper cables connectivity uses BASE-T PHYs such as 1000BASE-T and 10GBASE-T. Another type of copper cable is a twinaxial DAC. DAC connects via a serial transceiver or PHY using SFP, SFP+ or SFP28 for GE, 10GE and 25GE respectively. QSFP+ was specified in support of 40GE. QSFP28 supports 100GE. The SFP, SFP+, QSFP+, SFP28, and QSFP28 are all specified in the SFF specifications. The optical transceiver is specified by IEEE standards such as 1000BASE-SX, 10GBASE-SR, 40GBASE-SR4, and 100GBASE-SR4.
Internal PHY

Many controllers are a single ASIC with integrated MAC and PHY blocks to reduce the footprint required on the PCB.

External PHY

A NIC can be implemented via a MAC and an external PHY using chip-to-chip interfaces and serial management interfaces. The management interface is known as the Management Data Input/output (MDIO). It controls and manages the PHY devices (master side). This interface provides the MAC and software that monitors and controls the state of the PHY. The chip-to-chip interface is defined in the MII standard. The Media Independent Interface (MII, GMII, etc) is the link from the MAC to the PHY.

PHY Features

Auto-Negotiation

The IEEE Std 802.3-2015 defines several versions of auto-negotiation: Clause 28, 37, and 73. These clauses define a mechanism and state machine to negotiate the highest common link speed.

Forward Error Correction

Error detection and correction may be required depending on the technology. Two FEC versions are common—Reed Solomon and Fire Code. FEC adds additional bits to the data to enable the receiver to detect and correct some errors.

10GBASE-KR defines the serial PHY layer used across midplanes and connectors inside a chassis. FEC is an optional sub-layer of 802.3ap (Backplane Ethernet). The PHY uses auto-negotiation support to enable FEC on both sides of the link. The PHY transfers 10GBASE-R 64b/66b code words in FEC protected blocks. It is targeted at single burst error correction using a cyclic (2112, 2080) burst error correction code. FEC improves overall system reliability by significantly lowering the bit error rate.

As mentioned before, auto-negotiation is used to advertise FEC capabilities for a 10GBASE-KR PHY by using Clause 73 AN in the IEEE Std 802.3-2015 standard.

The parameters are advertised during Auto-Negotiation. FEC is enabled on the link only if both link partners advertise that they have FEC ability and one of them requests to enable FEC.

FEC enables a BER of $10^{-12}$ or better on a broader set of backplane channels. It improves overall system reliability and Mean Time to False Packet Acceptance (MTTFPA) requirements for 10GbE. FEC has a small latency penalty. It can correct burst errors of up to 11 bits over an NRZ channel via 2080 bits of payload and 32 bits of overhead.
Energy Efficient Ethernet

On an average, data centers’ Ethernet links are greatly underutilized and remain idle most of the time. However, the NICs always run idle pulses at wire speed, consuming non-productive power. This is the case for 1GE, 10GE, and higher speeds. Energy conservation is the motivation for Energy Efficient Ethernet (EEE).

EEE enables a Power Idle mode to disable part of the PHY on both sides of the link and save power during periods of low link utilization.

EEE is specified in IEEE Std 802.3az-2010 as a protocol to transition the PHY to a lower power consumption state in response to lack of traffic to be transmitted. It uses Low Power Idle (LPI) to save energy if the MAC/PHY supports EEE on both sides of the link.

The IEEE defines a mechanism to reduce power consumption during periods of low link utilization for the following PHYs:

- 100BASE-TX (Full Duplex)
- 1000BASE-T (Full Duplex)
- 10GBASE-T
- 10GBASE-KR
- 10GBASE-KX4
- 1000BASE-KX

It defines a protocol to coordinate transitions to idle power during periods when there is no data to be transmitted with the following objectives:

- The link status should not change as a result of the transition
- No frames in transit should be dropped or corrupted during the transition
- The transition should be transparent to upper layer protocols and applications

![Figure 33. EEE States](image-url)
During periods where no packets are transmitted, the MAC sends LP_IDLE to the PHY, putting it into a low-power state.

Refresh and alert schemes to keep channel information fresh and return quickly to activity.

Normal IDLEs indicate transition back to full link rate.

**Figure 34.** EEE in the Stack

During periods where no packets are transmitted, the MAC sends LP_IDLE to the PHY. The PHY goes into low power state on both sides.

Figure 35 illustrates how EEE works. This is to show the interaction between both sides of the link between the NIC and a switch when the transmitter on the NIC asserts or de-asserts LPI.

**Figure 35.** EEE Operation
The local PHY generates Assert LPI to signal the link partner that the local transmitter is entering sleep mode. During this time, the PHY Transmit function periodically sends refresh signals that are used by remote receiver to update adaptive filters. When there is traffic to send, the local PHY sends a Wake signal to bring the remote PHY to an operating state.

The default behavior of EEE can be tuned by using Link Layer Discovery Protocol (LLDP) to change wake times. The longer the wake time, the longer the delay until frames can pass providing a trade-off between energy savings and latency.

**Physical Communication Channel**

The network or fabric is constantly evolving as depicted in Figure 36. The diagram shows link speed evolution across various technologies such as Ethernet, Fibre Channel, and InfiniBand. Single lane speeds for Ethernet have increased from 1 Gbps to 10 Gbps and from 25 Gbps to 50 Gbps. Advanced encoding techniques such as PAM4 make high speeds possible by running at the same frequency supported by existing communication links. Techniques such as FEC enable acceptable bit error rates over noisy channels.

![Figure 36. Link Speed Transitions](image-url)
Copper Channels

Ethernet twisted pair copper cable BASE-T Ethernet technologies have continued to evolve since its inception as 10BASE-T (10 Mbps) in 1990. With advances in DSP techniques, such as encoding and error correction codes together with higher quality cables, BASE-T supports speeds from 10 Mbps up to 10 Gbps over Unshielded Twisted Pair (UTP) copper cable. This rate of increasing link speeds is greater than 1000x in less than 30 years. Higher speed cables are required to support 25GBASE-T and 40GBASE-T over shorter cable distances (30 m vs. 100 m).

![Supported Cables Diagram](image-url)

The other copper cable is a twinax cable (DAC) that includes support for SFP supporting 1 Gbps, SFP+ for 10 Gbps, and SFP28 for 25 Gbps. Greater speeds can be achieved by multi-lane composition for 4x10G (40G) using QSFP cables. 4x25G with QSFP28 are used for a 100 Gbps link. DAC supports up to 7 m cables. Longer distances can be supported up to 100 m via UTP copper cable. DAC and BASE-T cables support Top of Rack, End of Row, and Middle of Rack switch to server interconnects. BASE-T supports structured cable infrastructure where a server farm is connected to a switch fabric via a patch panel.
25GBASE-T, a 25 Gb/s standard over twisted pair, is being developed alongside 40GBASE-T within IEEE Std 802.3bq-2016 as of the writing of this book. This supports up to 30 m of CAT8 shielded twisted pair (STP) copper cables.

**Optical Channels**

The NIC supports optical channels when longer links are required using fiber instead of copper cable. There are two main optical cables used: MMF and SMF. SMF is used for long range connectivity. On the other hand, MMF is used for shorter range. The most popular optical deployment in the data center is MMF with an LC connector housing an SR optical transceiver.
Connector Types

Table 12 summarizes the key connector form factors. The small form-factor pluggable (SFP) supports optical transceivers and Direct Attached Copper cables for 1 Gbps, 10 Gbps, and 25 Gbps links. The Quad SFP supports transceivers for 40, 50, 100 and 200 Gbps. It supports up to four channels that can be used to implement 40GE as 4 x 10 Gbps using Multi-lane distribution (MLD). Similarly, 100GE is 4 x 25 Gbps and 200GE is 4 x 50 Gbps. RJ-45 is used with twisted pair copper wiring.

Table 12. Connector Types

<table>
<thead>
<tr>
<th>Link</th>
<th>Twisted Pair</th>
<th>Single Lane</th>
<th>Four Lanes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1G</td>
<td>RJ-45</td>
<td>SFP</td>
<td></td>
</tr>
<tr>
<td>10G</td>
<td>RJ-45</td>
<td>SFP+</td>
<td></td>
</tr>
<tr>
<td>25G</td>
<td>RJ-45</td>
<td>SFP28</td>
<td></td>
</tr>
<tr>
<td>40G</td>
<td>RJ-45</td>
<td>QSFP+</td>
<td></td>
</tr>
<tr>
<td>50G</td>
<td></td>
<td>QSFP28</td>
<td></td>
</tr>
<tr>
<td>100G</td>
<td></td>
<td>QSFP28</td>
<td></td>
</tr>
<tr>
<td>200G</td>
<td></td>
<td>QSFP56</td>
<td></td>
</tr>
</tbody>
</table>

Channel Types

Different types of cables are available based on the distance and bandwidth requirements. There are copper cables such as unshielded twisted pair, shielded twisted pair, and twinax. Optical cables include multi-mode fiber and single-mode fiber.

CAT5/6 UTP

This is a low cost CAT5/6 cable made of unshielded twisted pairs of copper wires bundled together that reach up to 100 m for 1000BASE-T. CAT6A is required for 100 m of 10GBASE-T over RJ-45 connector.

Figure 39. UTP Cable
CAT7 STP

As link speeds increase from 10 Gbps to 25 Gbps or higher, a BASE-T Shielded twisted pair is required. Each pair of wires is shielded and the whole bundle is shielded to minimize channel impairments such as noise and cross talk.

NOTE: IEEE Std 802.3bq-2016 specifies CAT7E and CAT8.

DAC

DAC is a twinax copper cable for direct connect to a ToR switch. It supports 1 Gbps–100 Gbps using SFP, SFP+, SFP28, QSFP+, and QSFP28 connectors.

DAC cables come as a Passive Copper Cable, such as the 10GE SFP+ DAC cable. It is a transparent cable between the NIC and a switch. A passive cable passes the signal through it without any signal conversion, amplification, or equalization. These cables can only support shorter distances in the order of 5–7 m suitable for intra-rack connectivity.

On the other hand, there are Active DAC SFP+ cables with integrated signal processing circuitry in the connectors that perform signal conversion (electrical-optical), amplification, or equalization. These cables can only support longer distances in the order of 10–15 m and are suitable for inter-rack connectivity.
Fiber

Optical signals use light pulses that are carried over fiber cables made out of glass or plastic strands. There are MMF and SMF cables for longer distances typically over an LC connector. Most NICs connect via MMF cables using an SFP+ module with an LC connector.

![Fiber Optic Cable](image1)

**Figure 43.** Fiber Optic Cable

![LC Connector](image2)

**Figure 44.** LC Connector

Backplane/Midplane

In modular servers, the mezzanine PHY connects to the IO module via traces and connectors on a printed circuit board up to 1 m. The PHY is typically integrated in the MAC on the mezzanine card. It runs using a KR SerDes in most cases on a 10GE port.
System boot is the process of loading an operating system typically stored in local persistent storage—HDD or SSD. There are also mechanisms to boot over the LAN or SAN network.

Network booting is accomplished via Program Download or Disk Emulation. Program Download, as the name indicates, downloads an executable file over the network for execution on the target server. This boot strap program uses the network interface to download the operating system. Preboot Execution Environment (PXE) is an example of Program Download. Once downloaded, the OS is stored on a local disk.

Disk Emulation uses the code in OptionROM to emulate a physical disk. The operating system executes on the remote target. Reads and writes to the disk are sent via the network. Examples of Disk Emulation include FCoE and iSCSI boot.

The NIC OptionROM includes the network stack code required for PXE, iSCSI, and FCoE boot.

The NIC ROM replaces the BIOS service routines for either the disk services (INT13) or boot vector (INT19) to emulate a disk.

**Option ROM**

An Option ROM (Read Only Memory) is a software component located in a ROM chip on an add-in card or the system board. Its physical address is in the system memory between addresses C0000h and DFFFFh. The BIOS copies this component to shadow memory during POST. An Option ROM is characterized by the first two locations containing a two-byte signature of 55h AAh. Option ROMs are responsible for initializing their associated hardware, allowing them to be available to the rest of the system for booting or runtime.
PXE Boot

PXE boot uses a program download mechanism through a network interface to download and execute a program without using the local disk storage space.

PXE allows a system to boot from an operating system image downloaded over the network. By definition, PXE is invoked before an operating system is loaded, so there is no opportunity for the intermediate driver to load and enable a team. As a result, teaming is not supported as a PXE client. However, an individual physical port adapter that participates in a team when the operating system is loaded may be used as a PXE client.

While a teamed virtual adapter cannot be used as a PXE client, it can be used for a PXE server, which provides operating system images to PXE clients using a combination of Dynamic Host Control Protocol (DHCP) and the Trivial File Transfer Protocol (TFTP). Both of these protocols operate over IP and are supported by all teaming modes. Figure 45 shows an implementation example:

**Figure 45.** PXE Boot
There are a variety of ways to implement PXE: OptionROM, uEFI, and CD/USB Key with iPXE. iPXE is the leading open source network boot firmware. It provides a full PXE implementation enhanced with additional features such as:

- Boot from a web server via HTTP
- Boot from an iSCSI SAN
- Boot from a Fibre Channel SAN via FCoE

You can use iPXE to replace the existing PXE ROM on your network card, or you can chain load into iPXE to obtain the features of iPXE without the hassle of reflashing.

**iSCSI Boot**

iSCSI boot uses a mechanism called Disk Emulation. Instead of downloading an executable program through a network interface, iSCSI clients or initiators send SCSI commands to a SCSI storage device or target on a remote server and treat the remote storage device as a locally attached storage. iSCSI interactions are used to load an executable program such as an operating system to the memory of the server.

iSCSI boot enables diskless blades or diskless servers to boot from the central image location on the SAN.

iSCSI boot allows a server to connect to a remote disk over the network at boot time using the iSCSI protocol. The remote disk appears as a local drive. The server will be able to boot from the remote disk.

The boot process will be handled by an expansion boot ROM stored in the NIC EEPROM enabled as a boot device. It provides access to the remote iSCSI target as if it were a local disk.

The iSCSI boot ROM accesses the network using the Universal Network Driver Interface (UNDI). The iSCSI boot ROM will have its own Transmission Control Protocol (TCP) implementation and will use the UNDI APIs in the same manner that the current packet drivers are being used.

The iSCSI initiator and target configurations are stored in the EEPROM including the Challenge-Handshake Authentication Protocol (CHAP) username and password and option ROM enabled/disabled for this port. iSCSI Option ROM reads these configurations and builds an iSCSI Boot Firmware Table (iBFT). The iBFT structure is created as per Microsoft’s iBFT specification. For more details, refer to the latest Microsoft iBFT specification.

The iBFT is the mechanism by which configuration settings are passed to the OS Initiator. This is necessary to maintain the iSCSI session that is established pre-OS boot. These specifications were initially developed and proposed by Microsoft and are expected to be part of ACPI specifications.
The OS iSCSI initiator takes control of disk access. The iSCSI session created by the boot ROM is dropped. The iBFT defines the iSCSI Configuration Block parameters used to handoff the iSCSI connection to the OS Initiator.

Two different iSCSI initiators are being used to establish iSCSI sessions with the same target. The first initiator is loaded by the system at boot time and is called the ‘boot initiator’. The second initiator is loaded by the OS loader and it is called the ‘OS initiator’. The OS initiator is responsible for all disk access to the target throughout the runtime of the operating system.

![iSCSI Boot via OptionROM - Before OS Boots](image)

**Figure 46.** iSCSI Boot Process

**Fibre Channel over Ethernet Boot**

FCoE Boot enables the user to boot an operating system from a remote target.

In the legacy BIOS boot mode, the NIC owns all aspects of networking and network booting. The NIC vendor’s setup pages control which features are enabled and how they work.

In the UEFI boot mode, the BIOS owns the networking stack, similar to the operating system.

Some CNA/HBA NICs can be configured to emulate a disk controller instead of, or in addition to providing a NIC interface. The BIOS considers the emulated disks as any other RAID or disk controller. The emulation settings are controlled by the NIC’s setup pages.
**UEFI Boot**

The Unified Extensible Firmware Interface (UEFI) BIOS replaces the legacy BIOS. It defines the interfaces between the platform, add-in cards, and the OS. It also provides the software stack to support network boot such as PXE and iSCSI. The NIC device OptionROM is not used by UEFI. UEFI includes the drivers in the system flash in place of option ROM.

The UEFI specification implements a TCP/IP network stack available before the OS boots. It includes VLAN support and PXE booting UEFI operating systems over a network. The UEFI network stack is part of the BIOS instead of the NIC’s ROM.

NICs include a legacy ROM image and a UEFI driver in their ROMs. The BIOS automatically scans for and loads the UEFI driver for each card if the BIOS is set to UEFI mode.

To provide device-independent support for network interfaces from multiple vendors, BIOS has recently added an iSCSI boot feature in the UEFI boot mode. In legacy BIOS, iSCSI boot was implemented via NIC firmware in the NIC device OptionROM. The iSCSI boot feature was device and vendor specific based on the invocation of an iSCSI boot stack that is provided by the PCI option ROM of the network interface. This form of iSCSI boot is supported in the legacy BIOS mode of the server.
The software elements required to operate the NIC include OS network stack and libraries, device driver, and firmware. Advanced drivers enable features such as NIC teaming.

**Base Driver**

The base driver is a software that controls the LAN controller hardware. It allocates resources for buffer descriptors and data buffers for Tx and Rx. The Transmit function copies frames to its buffer space and decides how to process the frame and place packets on the wire. The receive function implements the control and configuration necessary to process incoming packets. It also implements deferred procedure calls to process interrupts. It complies with each OS requirement such as NDIS for Windows and NetDev for Linux. NetDev defines network device parameters, which are specific to the driver used by the network device.

![Software Stack Diagram](image-url)
The primary functions of a NIC driver are:

- Adapter initialization
- Adapter reset
- Adapter shutdown
- Packet reception
- Packet transmission
- Interrupt handling

**NIC Teaming**

Single-homed servers (with one interface to the network) represent a single point of failure that significantly affects overall server availability. Enterprises use several approaches to improve reliability. The most common is to provide redundancy for the critical components in a server.

Multiple NICs in servers offer the benefit of traffic segmentation and failure isolation. However, this configuration is not fault tolerant and cannot scale easily. In addition, no bandwidth is available beyond what each NIC can provide. To use multiple network ports in a server more effectively, enterprises can create a logical or virtual adapter by grouping multiple physical adapters linked by an intermediate driver. The software stack in the OS treats such teams as one logical adapter. If a link or physical adapter fails, traffic dynamically rebalances over the remaining adapters on a load-balancing team or shifts from the primary adapter to the secondary adapter for a fault tolerant team.

Clearly, high availability and high performance in the IT infrastructure are mandatory. The IT strategies for a highly available network include network link aggregation, load balancing, and fault tolerance (FT), especially on the server. Link aggregation scales the available bandwidth by grouping multiple physical links together to form a single logical or virtual link. The redundant links provide both load balancing and fault tolerance. Traffic flows are redirected around a failed NIC or cable without interrupting applications.

The concept of grouping multiple physical devices to provide load balancing and fault tolerance is not new. Storage devices use RAID technology to group individual hard drives. Switch ports can be grouped using technologies such as IEEE Std 802.3ad Link Aggregation. Network interfaces on servers can be grouped into a team of physical ports called a virtual adapter.

Teaming is implemented by third parties via an NDIS intermediate driver in the Windows operating systems environment. This software component works with the miniport base driver, the NDIS layer, and the protocol stack to enable the teaming architecture.
The miniport driver controls the host LAN controller directly to enable functions such as send, receive, and interrupt processing. The intermediate driver fits between the miniport driver and the protocol layer, multiplexing several miniport driver instances and creating a virtual adapter that looks like a single adapter to the NDIS layer. NDIS provides a set of library functions to enable the communications between either miniport drivers or intermediate drivers and the protocol stack. The protocol stack implements IP, IPX, and ARP. A protocol address such as an IP address is assigned to each miniport device instance. However, when an intermediate driver is installed, the protocol address is assigned to the virtual team adapter and not to the individual miniport devices that make up the team.

The intermediate driver continually monitors the physical ports in a team for link loss. In the event of link loss on any port, traffic is automatically diverted to other ports in the team. The load balancing teaming mode supports switch fault tolerance by allowing teaming across different switches. However, for switch fault tolerance to happen, the switches should be on the same physical network or broadcast domain and should be stacked.

The primary function of the teaming driver is to provide fault tolerance and load balance inbound and outbound traffic among the networking ports installed on the system and grouped for teaming. The inbound and outbound load balancing algorithms are independent of each other. The outbound traffic for a particular layer 3 connection can be assigned to a given port while its corresponding inbound traffic can be assigned to a different port. The teaming driver also provides VLAN tagging support through its virtual networking interface, which may be bound to a single stand-alone port or to a team of ports.

For the purposes of this book, the network stack in an OS can be considered to consist of the protocol stack (e.g. TCP/IP, IPX, etc.), a base driver that understands the underlying hardware, and the networking device or port.

Generally, an application talks to the protocol stack, then the protocol stack talks to the base driver, which in turn talks to the networking device. Usually, there is some glue provided by the OS to manage the three pieces. The teaming driver is implemented as an intermediate driver. It operates below protocol stacks such as TCP/IP and IPX and exposes a virtual networking interface to the protocol layers. This virtual networking interface uses the MAC Address of a chosen port in the team. A Layer 3 address must also be configured for the virtual network interface.

When the team interface is initialized in the system, it binds itself to the TCP/IP protocol. Hence, it may either obtain a DHCP IP address or be assigned a static IP address by the user. The team members’ bindings to TCP/IP are removed, and they are now bound to the teaming driver protocol edge that the intermediate driver exposes. Hence, the ports are not seen by the TCP/IP layer in the OS and cannot obtain/be assigned an IP address. In addition, the teaming driver associates the MAC address of the primary port in the team with the team’s TCP/IP address.
If the primary network connection is an adapter and it is hot-plugged out of the server and that adapter is inserted somewhere else in the network, the MAC address will appear on the network in two places unless the server is rebooted. On reboot, the team will use the MAC address of the new primary port.

Network Addressing

To understand how teaming works, it is important to understand how node communications work in an Ethernet network. The following information provides a high level overview of the concepts of network addressing used in an Ethernet network.

Every Ethernet network interface in a host platform, such as a server, requires a globally unique Layer 2 address and at least one globally unique Layer 3 address. Note that a layer 3 address is only a requirement when communicating by using a layer 3 protocol over a global public network (and without the aid of an address translating service). Layer 2 is the Data Link Layer and Layer 3 is the Network layer as defined in the OSI model. The Layer 2 address is assigned to the hardware and is often referred to as the MAC address or physical address. This address is pre-programmed at the factory and stored in NVRAM on a NIC or on the system motherboard for an embedded LAN interface. In addition, the burnt-in MAC address can be modified via a locally administered address mechanism in the OS driver.

The layer 3 addresses are referred to as the protocol or logical addresses assigned to the software stack. IP and IPX are examples of Layer 3 protocols. In addition, Layer 4 (Transport Layer), such as TCP and UDP, uses port numbers for each network upper level protocol, such as Telnet or FTP. These port numbers are used to differentiate traffic flows across applications. Layer 4 protocols such as TCP or UDP are most commonly used in today’s networks. The combination of the IP address and the TCP port number is called a socket.

Ethernet devices communicate with other Ethernet devices by using the MAC address and not the IP address. However, most applications work with a host name that is translated to an IP address by a Naming Service such as WINS and DNS. Therefore, a method of identifying the MAC address assigned to the IP address is required. The ARP for an IPv4 network or neighbor discovery for IPv6 provides this mechanism. For IPX, the MAC address is part of the network address and ARP is not required. ARP is implemented using an ARP Request and ARP Reply frame. ARP Requests are typically sent to a broadcast address while the ARP Reply is typically sent as unicast traffic. A unicast address corresponds to a single MAC address or a single IP address while a broadcast address is sent to all devices on a network.

Teaming and Network Addresses

There are many different teaming modes implemented by the hardware and OS vendors. This book describes only the fundamental concepts and does not cover all the teaming modes. A team of adapters function as a single virtual network interface. A virtual network adapter advertises a single layer 2 and one or more layer 3 addresses.
When the teaming driver initializes, it selects one MAC address from one of the physical adapters to be the Team MAC address. This address is typically taken from the first adapter that gets initialized by the driver. When the server hosting the team receives an ARP Request, it will select one MAC address from the physical adapters in the team to be used as the source MAC address in the ARP Reply.

In Windows operating systems, the IPCONFIG/all command shows the IP and MAC addresses of the virtual adapter. The protocol IP address is assigned to the virtual network interface and not to the individual physical adapters.

For switch independent teaming modes, all physical adapters that constitute a virtual adapter must use the unique MAC address assigned to them when transmitting data. That is, the frames that are sent by each of the physical adapters in the team must use a unique MAC address to be IEEE compliant. It is important to note that ARP cache entries are not learned from the received frames, but only from ARP Requests and ARP Replies.

**Link Status**

Link status allows the teaming driver to determine if a port is in an active state and is able to communicate on the network. The base drivers for the ports that are part of the team detect link status for the hardware and notify the upper layers of the stack when link-up and link-down events occur. These events are received by the teaming driver via the NDIS wrapper. This indication is used as a direct stimulus for the intermediate driver to perform failover if the link down event is associated with the primary/active port in the team.

It is important to note that link status detection only occurs between the ports in the team and their immediate link. Typically, NIC teaming has no way of reacting to other hardware failures in the switches and cannot detect loss of link on other ports.

**Description of Teaming Modes**

There are three methods for classifying the supported teaming modes, which are based on whether:

- The switch port and the NIC support teaming
- The functionality of the team supports load balancing and failover or just failover
- The Link Aggregation Control Protocol (LACP) is used or not

**Fault Tolerance**

An FT team provides basic redundancy functionality and can be configured with two or more physical ports.

An intrinsic primary port is chosen by the teaming driver (based on the speed and capabilities of the ports that are part of the team) and is the only port in the team that is Active and transmits or receives clients’ data.
Secondary ports are in Standby mode, ready to replace the primary port if it fails. If the primary port fails for any reason (e.g., port hardware failure, cable loss, switch/hub fault, etc.), a secondary port assumes the properties of the primary port, and is made Active. The new primary port will continue the communication with clients. This process is transparent to the clients and the user application. Ports may be in a Disabled, Standby, or Active state. A disabled state for a port indicates that this port is not functioning as part of the team and will not be used for failover. A disabled state for a primary port indicates that failover is imminent to a standby port.

FT uses four indicators to detect if a failover is needed: the primary’s link status, the primary’s hardware status, a probe mechanism between the members of the team, and the primary port’s packet receive counters.

When a failover occurs, the load is transferred from the primary to the secondary fast enough to prevent user protocol sessions (e.g., TCP, file transfers) from being disconnected. After failover to the secondary, if the user selects a preferred primary port in the team, once this preferred primary port’s link is restored, all connections are restored to the preferred primary port. To achieve this fail-back, the preferred primary is again made Active, and the secondary is returned to Standby status.

In this team mode, all the ports in the team use the MAC address of the primary port. The port which is currently active and transmitting in the team uses the primary MAC as the source MAC address in the Ethernet header of the transmitted packet. This ensures that the corresponding switch port learns the primary MAC address. This also ensures transparency in failover to a secondary port.

**Switch Fault Tolerance**

A Switch Fault Tolerant team provides fault tolerance between ports connected to two different switches. This team can be configured with a maximum of two ports. In such a configuration, one of the ports connected to one of the switches is the Active connection. The other port is in Standby mode. The Active port may be the intrinsic primary chosen by the intermediate driver, or the preferred primary port selected by the user. When the Active port loses link, the teaming driver will use the secondary port connected to the second switch as backup and activate it.

In this mode, since one switch is redundant, both switches must be cross connected. When the primary switch fails, all active connections can be forwarded through the redundant switch. Hence, in this case STP must be enabled on all switches in the network to resolve loop-back link states. However, the switch ports connected to the server teams should have Port Fast or Edge Port enabled.

If the user chooses a preferred primary port in the team and failover occurs to the secondary port, when the switch port connected to the preferred primary is functional again, the intermediate driver handles fail-back of connections just like in FT. However, a time delay of 60 seconds is introduced here to allow for STP convergence on the switch network. This is to prevent packet-loss that could occur if the preferred primary port is activated before its switch port becomes fully active.
In recent years, many alternatives to enhance or replace STP have been developed. Some of them are:

- Transparent Interconnection of Lots of Links (TRILL)
- Shortest Path Bridging (SPB)
- Virtual Link Trunking (VLT)
- Multi-chassis Link Aggregation (MLAG)
- Virtual Port Channel (vPC)

**Host-Based Load Balancing**

Load Balancing provides both load balancing and failover while fault tolerance supports only failover. It works with any Ethernet switch and requires no trunking configuration on the switch. The team advertises multiple MAC addresses and one or more IP addresses (when using secondary IP addresses). The team MAC address is selected from the list of load balancing members. When the server receives an ARP Request, the software networking stack will always send an ARP Reply with the team MAC address. To begin the load balancing process, the teaming driver will modify this ARP Reply by changing the source MAC address to match one of the physical adapters.

Host-based Load Balancing enables both transmit and receive load balancing based on the Layer 3/Layer 4 IP address and TCP/UDP port number. In other words, the load balancing is not done at a byte or frame level but on a TCP/UDP session basis. This methodology is required to maintain in-order delivery of frames that belong to the same socket conversation. In practice, load balancing is supported on 2–8 ports. These ports can include any combination of add-in adapters and LAN-on-Motherboard (LOM) devices. Transmit load balancing is achieved by creating a hashing table using the source and destination IP addresses and TCP/UDP port numbers. The same combination of source and destination IP addresses and TCP/UDP port numbers will generally yield the same hash index and therefore point to the same port in the team. When a port is selected to carry all the frames of a given socket connection, the unique MAC address of the physical adapter is included in the frame, and not the team MAC address. This is required to comply with the IEEE Std 802.3-2015 standard. If two adapters transmit using the same MAC address, then a duplicate MAC address situation would occur that the switch cannot handle gracefully.

Receive load balancing is achieved through an intermediate driver by sending Gratuitous ARPs (G-ARPs) on a client-by-client basis using the unicast address of each client as the destination address of the ARP Request (also known as a Directed ARP). This is considered client load balancing and not traffic load balancing. When the intermediate driver detects a significant load imbalance between the physical adapters in an HBLB team, it will generate G-ARPs in an effort to redistribute incoming frames. The intermediate driver does not answer ARP Requests; only the software protocol stack provides the required ARP Reply. It is important to understand that receive load balancing is a function of the number of clients that are connecting to the server via the team interface.
Receive load balancing attempts to load balance incoming traffic for client machines across physical ports in the team. It uses a modified G-ARP to advertise a different MAC address for the team IP address. This G-ARP is unicast with the MAC and IP address of a client machine in the target physical and protocol address respectively. This causes the target client to update its ARP cache with a new MAC address map to the team IP address. G-ARPs are not broadcast because this would cause all clients to send their traffic to the same port. As a result, the benefits achieved through client load balancing would be eliminated and could cause out of order frame delivery. This receive load balancing scheme works as long as all clients and the teamed server are on the same subnet or broadcast domain.

When the clients and the server are on different subnets, and the incoming traffic has to traverse a router, the received traffic destined for the server is not load balanced. The physical adapter that the intermediate driver has selected to carry the IP flow will carry all of the traffic. When the router needs to send a frame to the team IP address, it will broadcast an ARP Request (if not in the ARP cache). The server software stack will generate an ARP Reply with the team MAC address, but the intermediate driver will modify the ARP Reply and send it over a particular physical port.

The reason is that ARP is not a routable protocol. It does not have an IP header and therefore is not forwarded onto other networks by the router. ARP is only a local subnet protocol. In addition, since the G-ARP is not a broadcast packet, the router will not process it and will not update its own ARP cache.

The only way that the router would process an ARP that is intended for another network device is if it has Proxy ARP enabled and the host has no default gateway. This is very rare and not recommended for most applications.

Transmit traffic through a router will be load balanced as transmit load balancing is based on the source and destination IP address and TCP/UDP port number. Since routers do not alter the source and destination IP address, the load balancing algorithm works as intended.

**Switch Based Generic Trunking**

Generic Trunking is a switch-assisted teaming mode and requires configuring ports at both ends of the link: server interfaces and switch ports. This is often referred to as IEEE Std 802.3ad Link Aggregation static mode. In this mode, the team advertises one MAC Address and one IP address when the protocol stack responds to ARP Requests. In addition, each physical adapter in the team uses the same team MAC address when transmitting frames. This is possible since the switch at the other end of the link is aware of the teaming mode and will handle the use of a single MAC address by every port in the team. The forwarding table in the switch will reflect the trunk as a single virtual port.

In this teaming mode, the intermediate driver controls load balancing and failover only for outgoing traffic while incoming traffic is controlled by the switch firmware and hardware. As is the case for Host Based Load Balancing, the intermediate driver uses the IP/TCP/UDP source and destination addresses to load balance the transmit traffic from
the server. Most switches implement an XOR hashing of the source and destination MAC address.

**Link Aggregation (IEEE Std 802.3ad LACP)**

Link Aggregation is similar to Generic Trunking except that it uses the LACP to negotiate the ports that will constitute the team. LACP must be enabled at both ends of the link for the team to be operational. If LACP is not available at both ends of the link, 802.3ad provides a manual aggregation that only requires both ends of the link to be in a link up state. Because manual aggregation activates a member link without performing the LACP message exchanges, it should not be considered as reliable and robust as an LACP negotiated link. LACP automatically determines which member links can be aggregated and then aggregates them. It provides for the controlled addition and removal of physical links for the link aggregation so that no frames are lost or duplicated. The removal of aggregate link members is provided by the marker protocol that can be optionally enabled for LACP enabled aggregate links.

The Link Aggregation group advertises a single MAC address for all the ports in the trunk. The MAC address of the aggregator can be the MAC addresses of one of the MACs that make up the group. LACP and marker protocols use a multicast destination address such as the reserved multicast 01-80-C2-00-00-02.

The Link Aggregation control function determines which links may be aggregated and then binds the ports to an aggregator function in the system and monitors conditions to determine if a change in the aggregation group is required.

Link aggregation combines the individual capacity of multiple links to form a high performance virtual link. The failure or replacement of a link in an LACP trunk will not cause loss of connectivity. The traffic will simply be failed over to the remaining links in the trunk.

**OS Based Teaming**

Historically, NIC teaming has been provided via the NIC hardware vendors for Windows and Linux. However, NIC teaming is now part of the OS kernel. Linux implements channel bonding, VMware vSphere provides NIC teaming as part of the hypervisor virtual switch (vSwitch), and in Windows Server it is called LBFO and SET provided in Windows Server 2016. The software component to create and manage the NIC teaming is built into the OS.

Earlier, one could not mix NICs in a team from two different manufacturers or at least not easily and with limitations. This made it difficult to add or replace a NIC if the original was not available or difficult to find. Also, if you got a newer NIC, you would have to check the compatibility with the NIC teaming software before adding it.

OS based teaming supports mixing NICs from different manufacturers and new NICs can be added or replaced without checking for compatibility with the existing teaming software.
**Virtual LANs**

The term VLAN (Virtual Local Area Network) refers to a collection of devices that communicate as if they were on the same physical LAN. Any set of ports (NIC and Switch) can be considered a VLAN. LAN segments are not restricted by the hardware that physically connects them.

In 1998, the IEEE approved the 802.3ac standard, which defines frame format extensions to support Virtual Bridged Local Area Network tagging on Ethernet networks as specified in the IEEE Std 802.1Q specification. The VLAN protocol permits insertion of a tag into an Ethernet frame to identify the VLAN to which a frame belongs. If present, the 4-byte VLAN tag is inserted into the Ethernet frame between the source MAC address and the length/type field. The first two-bytes of the VLAN tag consist of the 802.1Q tag type while the second two bytes include a user priority field and the VLAN ID.

VLANs allow the user to split the physical LAN into logical networks. Each defined VLAN behaves as its own separate network, with its traffic and broadcasts isolated from the others, thus increasing bandwidth efficiency within each logical group. VLANs also enable the administrator to enforce appropriate security and Quality of Service (QoS) policies. In some cases, the NIC teaming driver also supports the creation of multiple VLANs per team or adapter. In some operating systems environments, VLANs and teams are treated separately, handled by separate drivers. However, the operating system and system resources limit the actual number of VLANs. VLAN support is provided according to IEEE Std 802.1Q and is supported in a teaming environment as well as on a single adapter.

The software support required to enable and configure VLANs is often part of the NIC teaming. The teaming intermediate driver supports VLAN tagging. One or more VLANs may be bound to a single instance of the intermediate driver.

![VLAN Diagram](image)

**Figure 48.** VLANs

VLANs can group computers into logical workgroups. This can simplify network administration when connecting clients to servers that are geographically dispersed.

Typically, VLANs consist of co-workers within the same department but in different locations, groups of users running the same network protocol, or a cross-functional team working on a joint project.
VLANs provide the following benefits:

- Improved network performance
- Limited broadcast storms
- Improved LAN configuration updates (adds, moves, and changes)
- Minimized security problems
- Priority-based QoS

To set up IEEE VLAN membership (multiple VLANs), the host port must be attached to a switch or another host with IEEE Std 802.1Q VLAN capability. In most environments, a maximum of 64 VLANs per port can be set up, but the limit is implementation specific.

VLAN tagging is done according to the IEEE Std 802.1Q protocol for both single port and a team. Multiple VLANs can be configured over a single port or a team of ports up to a maximum of 64 VLANs. Each VLAN is represented by a virtual network interface that is bound to the upper level protocols on the host network stack.

![Figure 49. VLANs Example](image)

In some implementations, the teaming driver or VLAN service forwards the tagging information to the NIC hardware and relies on the hardware to insert the VLAN tag into the transmitted packet. The tags on received packets are also removed by the hardware. The tagging information is forwarded to the teaming driver or VLAN service, which ultimately forwards it to the protocol stack along with the packet. The switch port connected to the server NIC must be configured as a trunk port. This is required to allow frames with VLAN tagging between the switch and the server.
After an application sends data across a network, several data movement and protocol processing steps occur. These and other networking activities consume critical host resources (such as CPU cycles) during the execution of the following tasks:

- The application writes the transmit data to the TCP/IP sockets interface for transmission in payload sizes ranging from 4 KB to 64 KB.
- The OS segments the data into MTU–size packets and then adds TCP/IP header information to each packet.
- The OS copies the data onto the NIC send queue.
- The NIC performs the DMA transfer of each data packet from the TCP buffer space to the NIC and interrupts CPU activities to indicate completion of the transfer.

Instead of consuming CPU cycles on data movement, the NIC hardware performs more compute intensive operations. The two most popular methods to reduce the substantial CPU overhead that TCP/IP processing incurs are: TCP/IP checksum offload (CSO) and large send offload (LSO).

NIC offloads have come a long way over the last 10 years. There are two types of NIC offloads: stateful and stateless. The stateful offloads include Internet Protocol Security (IPSec) offload, iSCSI offload, TOE, and FCoE CNAs to name a few, and will be covered later in this book. The stateless offloads are described in this chapter.

**Checksum Offload**

The TCP/IP CSO technique moves the calculation of the TCP and IP checksum packets from the host CPU to the network adapter. For the TCP checksum, the transport layer on the host calculates the TCP pseudo-header checksum and places this value in the checksum field, thus enabling the network adapter to calculate the correct TCP checksum without touching the IP header. However, this approach yields only a modest reduction in CPU utilization.
CSO is a feature provided by many network adapters that allows the TCP/IP/UDP checksums for send and receive traffic to be calculated by the NIC hardware rather than by the host CPU. CSO reduces the server host CPU utilization. This frees up CPU cycles for application execution. A NIC that supports CSO will advertise this capability to the operating system so that the checksum does not need to be calculated in the protocol stack.

**Large send offload**

Large send offload (LSO), also known as TCP segmentation offload (TSO), frees the OS from the task of segmenting the application's transmit data into MTU-size chunks. Using LSO, TCP can transmit a chunk of data larger than the MTU size to the network adapter.

LSO is an extremely useful technology to scale performance across multiple Gigabit or higher speed Ethernet links. The LSO technique is most efficient when transferring large messages. LSO yields performance benefits only for traffic being sent; it offers no improvements for traffic being received. LSO has little effect on interrupt processing because it is a transmit-only offload. Methods such as TCP/IP CSO and LSO provide limited performance gains or are advantageous only under certain conditions. For example, LSO is less effective when transmitting several smaller-sized packages. Also, in environments where packets are frequently dropped and connections lost, connection setup and maintenance consume a significant proportion of the host’s processing power.

**Large Receive Offload/Receive Side Coalescing**

Large Receive Offload (LRO) also known as Receive Side Coalescing (RSC) is a feature provided by many network adapters that aid the host network stack in processing incoming network packets by aggregating them into fewer but larger packets. This is an advantage to the host as it requires less packet events to be handled by the host CPU and less processing per packet. This offloads the host CPU in a transparent fashion to the host network stack. It is the inverse operation to TSO/LSO on the transmit side.

**Receive Side Scaling**

A received packet from the network on a particular NIC will trigger an interrupt to the host processor eventually causing a Deferred Procedure Call (DPC) to be queued on one of the system processors. The DPC runs to completion on the processor that hosted the interrupt, and additional interrupts from the NICs are disabled until the DPC completes its cycle.
RSS is a feature provided by many network adapters that distribute the kernel-mode network processing load across all the cores in the CPU complex. RSS functions as an interrupt load balancer to ensure more efficient processing of the network traffic. It enables receive processing to be balanced across multiple processors in the system. RSS enables parallel DPCs and multiple interrupts concurrently.

RSS enables in-order packet delivery by ensuring that only one processor processes packets for a single TCP connection. The NIC uses a hashing function based on the header to compute a signature for the packet. The hash result is used as an index in an indirection table. The indirection table contains the CPU that will run the associated DPC. The RSS hash function can be based on the destination MAC or L2 through L4 port addresses.

**Virtual RSS**

On a virtualized server, RSS in the Guest VM distributes the network-processing load over multiple virtual processors by using hash and indirection tables. When virtual RSS is disabled, the maximum throughput for inter VM transfers is measured to be around 6 Gbps. When virtual RSS is enabled on the receiver VM, the throughput for VM-VM communication is measured to be around 8.5 Gbps.

**Virtual Machine Queues**

Virtual Machine Queues (VMQ) is an incredibly powerful performance enhancing technology introduced in Windows Server 2008R2 for Hyper-V. It allows assigning a base processor core to process a given VM network traffic. VMQ allows the NIC to offload the sorting and forwarding from the hypervisor vSwitch. Every VM has its own hardware queue that the NIC can use to transfer data to the VM instead of the vSwitch.

Setting the VMQ processing to Disabled forces all processing of VM network traffic to a single core, thereby limiting the throughput.

VMware’s vSphere implementation is called Net-Queue. Virtual Machine Queue or VMware’s Net-Queue is a feature provided by many network adapters that uses DMA to transfer incoming frames to the appropriate Virtual Machine receive queue. This offload is transparent to the virtual system and transfers packets directly to the shared memory in the target VM.

Transmitted data from the VM is sorted based on the destination MAC address and grouped. The data is sent to the vSwitch for queueing.

On receiving data, the NIC services the queues in a Round robin scheme to ensure fairness for all VMs.
Overlay network tunneling protocols are beneficial for large scale multi-tenant infrastructure. VMware supports VxLAN while Microsoft supports NVGRE and VxLAN.

Overlay tunneling was developed to solve the following limitations with the traditional VLAN approach in response to server virtualization and the cloud business model:

- 4096 VLANs are too few
- Multi-tenant data centers need overlapping MAC/VLANs
- Explosion of MAC table sizes at the ToR

VxLAN is an overlay tunnel running L2 over an L3 network where each overlay is a VxLAN segment. VMs must belong to the same VxLAN segment in order to communicate. The VxLAN header includes a 24-bit tag (VNI) to identify each segment or tunnel. Overlapping MAC addresses and VLANs can exist across tunnels.

The tunnel end point is implemented in the hypervisor. This is where the L2 frame (inner header) is encapsulated at the sending node and de-encapsulated (outer header) at the receiving node.

16 million VNIs may be defined in any administrative domain and each 24-bit VNI may contain up to 4,094 VLANs. Customer data is kept separate because only VMs operating within the same VNI tagged tunnel can communicate.
VxLAN Packet Offloads

Virtual Extended LANs are an overlay technology which incorporates an additional 24 bit VLAN identifier inside a UDP tunnel carrying the encapsulated payload. This expands the number of VLANs available at the endpoints of communication. The NIC is unable to perform the stateless offloads for encapsulated frames. The NIC requires hardware changes to add stateless offload support for VxLAN frames.

The following stateless offloads are supported for the VxLAN packet:

- Large Send Offload (LSO)
- Virtual Machine Queue (VMQ)
- Transmit (Tx) CSO (IPv4, TCP, UDP)
- Receive (Rx) CSO (IPv4, TCP, UDP)
- Receive Side Scaling (RSS)

\[\text{Figure 51. VxLAN Header}\]

NVGRE

Network Virtualization using Generic Routing Encapsulation (NVGRE) attempts to alleviate the scalability problems associated with large cloud computing deployments. It uses Generic Routing Encapsulation (GRE) to tunnel layer 2 packets over layer 3 networks. It includes a new 24-bit identifier, Tenant network ID (TNI), to identify each tunnel.

NVGRE uses overlay tunnels to carry L2 packets over the L3 network across subnet boundaries without application modification.

NVGRE allows for unicast, multicast, and broadcast traffic over the network. An NVGRE endpoint receives Ethernet packets from a VM and encapsulates and sends them through the GRE tunnel. The endpoint de-encapsulates incoming packets, distributing them to the proper VM. Endpoints can be located in any network component but is usually implemented in the server hypervisor.
Each NVGRE endpoint will be assigned at least one and possibly multiple IP addresses. Each IP address is referred to as a Provider Address. VM IP addresses are Customer Addresses. Assigning multiple Provider Addresses to an endpoint enables load balancing.

The NVGRE endpoint isolates individual tunnels by inserting the TNI specifier in the GRE header. The TNI identifier is placed in the key field specified by RFC 2890. The key field is defined as a 32-bit field, but the TNI occupies 24 bits. The remaining eight bits are reserved. NVGRE does not use the sequence number defined by the RFC 2890. The NVGRE endpoint encapsulating a packet must specify the destination address to which the packet should be sent.

Overlay tunnels enable multitenancy at cloud scale where a virtualized server can host VMs from multiple customers with overlapping VLANs. It also supports live migration across networks. An NVGRE-aware NIC supports the most common stateless offloads such as LSO, CSO, VMQ, and RSS on the encapsulated frame.

There will be NIC hardware support for:
- Stateless offloads for encapsulated frames
- Encapsulation

NDIS 6.30 (available in Windows Server 2012 and later) introduced NVGRE Task Offload, which makes it possible to use NVGRE-formatted packets with stateless offloads.

As of NDIS 6.30, NVGRE Task Offload feature does not specify the offloading of the encapsulation and de-encapsulation operations.
On the send path, the following task offloads are supported:

- Checksum computation of IPv4 and TCP or UDP payload
- LSO version 1 and LSO version 2

For send-side offloads, the miniport must perform corresponding operations on the tunnel (outer) IP header, the transport (inner) IP header, and the TCP header.

On the receive path, the following task offloads are supported:

- Checksum validation of IPv4 and TCP or UDP payload
- RSS
- VMQ

For receive-side offloads, the NIC must parse the encapsulation protocol headers. For example, for GRE encapsulation, the NIC must parse the GRE header and perform task offloads on the transport (inner) and/or tunnel (outer) IP headers.

The network adapter exposes this capability to the Hyper-V Host OS using the Encapsulated Task Offload property. The default setting is Enabled. Tests have shown that throughput on a 10GE NIC increases from around 4 Gbps to around 9 Gbps when using Encapsulated Task Offload.
Table 13 summarizes the I/O architectural options for virtualization of LAN I/O host devices. Emulated I/O is implemented via Type I and Type II devices. In traditional Emulated I/O mode, L2 switching is performed via the software vSwitch in the host hypervisor VM Manager (VMM) or DOM0. The inter-VM traffic stays inside the host platform as it is handled by the software vSwitch.

Direct I/O assignment is implemented using Type I, Type III, and Type IV devices. In Type I devices, direct I/O assignment is used to statically assign a physical function to a VM. In Type III and IV devices, direct I/O assignment is implemented by using an SR-IOV LAN controller device, where the physical function is shared across many VMs using virtual I/O functions. In this case, the L2 switching is implemented via a Virtual Ethernet Bridge (VEB) embedded hardware switch in the LAN controller device. Inter-VM traffic traverses the PCI bus twice, but it stays inside the host platform.

Another implementation of direct I/O with SR-IOV enables the VM-to-VM L2 switching at the external network switch versus the vSwitch. This is enabled via tagless or tagged mechanism as defined by Edge Virtual Bridge (EVB) and Bridge Port Extension (BPE) IEEE standards. In this case, the VM-to-VM traffic traverses outside the host platform for hair-pin mode switching on the external Top of Rack switch with EVB or BPE support.

Table 13. I/O Virtualization Architectures

<table>
<thead>
<tr>
<th>I/O Device Type</th>
<th>Description</th>
<th>NIC Virtualization</th>
<th>L2 Switching</th>
<th>PCI I/O Bus</th>
<th>External Network Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Standard Physical Device</td>
<td>Performed via hypervisor software</td>
<td>Performed via software vSwitch</td>
<td>Inter VM traffic does not go out on this link</td>
<td>Inter VM traffic does not go out on this link</td>
</tr>
<tr>
<td>II</td>
<td>Multiqueue</td>
<td>Performed via hypervisor software</td>
<td>Performed via software vSwitch</td>
<td>Inter VM traffic does not go out on this link</td>
<td>Inter VM traffic does not go out on this link</td>
</tr>
<tr>
<td>III</td>
<td>SR-IOV Mode w/VEB</td>
<td>Performed via PCIe configuration space</td>
<td>Performed via hardware e-switch in NIC</td>
<td>Inter VM traffic traverses twice</td>
<td>Inter VM traffic does not go out on this link</td>
</tr>
<tr>
<td>IV</td>
<td>SR-IOV Mode w/VEB or BPE</td>
<td>Performed via PCIe configuration space</td>
<td>Performed via external hardware switch</td>
<td>Inter VM traffic traverses twice</td>
<td>Inter VM traffic traverses twice</td>
</tr>
</tbody>
</table>
Emulated I/O

In Emulated I/O, the hypervisor VMM creates and manages vNICs for VM assignment. VMs do not access the physical NICs directly, but only see the emulated vNIC. The association of vNICs to the physical NIC is performed by a vSwitch implemented in the hypervisor software. The vSwitch is responsible for filtering and sorting of frames transferred between VMs and between VMs and the external network. Multi-queue support has been added to the I/O controller and hypervisors to assist the vSwitch with frame filtering and sorting.

CPU and chipsets have added enhancements that enable a device to DMA to/from host memory allowing VM direct assignment of hardware resources to improve performance. However, this is limited to the number of physical devices installed in the host platform. SR-IOV eliminates this limitation by the implementation of many VFs on physical ports for VM mapping.

Emulated I/O offers the following advantages:

- Common virtualized interface to all VMs
- Broad VM support with native drivers
- VMotion support
- Hardware independence

Emulated I/O has the following disadvantages:

- Software overhead limits performance
- Difficult to utilize the advantages of hardware offloads and innovations
- Software vSwitch and network switches are managed differently

Intra VM-to-VM network traffic on the same vSwitch does not go over any external physical NIC. Instead, the host vSwitch handles moving traffic between the VMs on the same host.

Figure 53. Virtualization Block Diagram
SR-IOV enables a natively shareable device that can bypass the hypervisor on a virtualized server for the main data movement. It enables a VM to have direct access to a PCI I/O device while sharing the device among multiple VMs or GOSes.

The PCI-SIG Single Root I/O Virtualization and Sharing specification defines a standard to implement a natively shareable device in a virtualized host. VFs provide independent DMA streams that can be assigned to a VM. A physical function includes the resource capabilities to create the VFs.

A physical PCIe function represents a single physical port. SR-IOV allows multiple VFs on a physical function that can be mapped to VMs. SR-IOV enabled hypervisors initialize and assign VFs to VMs.

SR-IOV With Internal Switching

The impetus for SR-IOV is the concept of virtualized hardware that can be shared natively and directly across VMs. An SR-IOV device can be shared across many VMs with no intervention of the hypervisor in the data path. Therefore, an embedded hardware switch is required to replace the hypervisor vSwitch to handle directly assigned VM traffic. As shown in Figure 55, SR-IOV eliminates the need for hypervisor in the data path.

The virtual soft switch is still required for legacy VMs, hypervisor kernel services (FT, NFS, iSCSI, Management, etc.), and VM mobility. The Single root PCI Manager function implemented by the hypervisor configures the end point device and assigns VF(s) to VM(s). Runtime data operations (DMA) are directly handled between the VF and VM.
SR-IOV requires the following:

- Support for PCI SR-IOV in the NIC
- Support for PCIe ECNs- ARI and FLR
- Framework management
  - VM Mobility must be built in from ground up
  - BIOS changes for platforms
  - VT-d and IOMMU support
  - PCIM integration in the hypervisor

Figure 55. SR-IOV With VEB

**SR-IOV With External Switching**

Historically, a physical server NIC port was connected to an external switch port. This is called the access layer of the network. Server virtualization modifies the network access layer when moving from physical to virtual servers. The hypervisor moves the access switch (vSwitch) to the inside of the host. The vSwitch is needed for VM-to-VM communications and VM communications over the LAN. This causes network visibility to end at the physical ToR external switch port for Inter-VM traffic. The management framework is also different. The physical network relies on switch IOS Web GUI or CLI while the vSwitches rely on the hypervisor management console.
SR-IOV devices bypass the vSwitch implemented by the hypervisor. A switch is embedded in the device to control Ethernet frame flow and packet replication. However, the Embedded switch (e-Switch) on SR-IOV devices implements inconsistent features and manageability as compared to the physical network switches.

The IEEE is working on two standards (IEEE 802.1Qbg-Edge Virtual Bridging and IEEE 802.1Qbh-Bridge Port Extension) to move the VM switching from the vSwitch or e-Switch to the external physical network switch. This moves the access layer back outside the server to the physical switch.

In the future, the SR-IOV device, as shown in Figure 56, will support external switching that provides external network visibility and management of all inter-VM traffic. The following are the key attributes of EVB and BPE:

- The component in the host hypervisor is no longer a virtual switch, but rather an EVB port aggregator or network interface virtualizer
- VM-to-VM forwarding is moved to an external switch
- Require a new switch port mode to allow frames to be forwarded back to the same port they were received from

![Figure 56. SR-IOV With External Switching](image-url)
Host Platform Requirements

The following hardware and software components are required to implement SR-IOV:

- **CPU**
  - Hardware virtualization: Intel’s Extended Page Table (EPT) or AMD’s Nested Page Table (NPT)

- **Chipset**
  - Interrupt Remapping: Intel’s Virtual Technology for Directed IO (VT-d) with the Interrupt Remapping capability (VT-d2) or any version of AMD’s I/O MMU
  - DMA Remapping: Intel’s VT-d with Queued Invalidations or any AMD I/O MMU
  - Access Control Services (ACS) on PCI Express root ports

- **BIOS**
  - Firmware tables which expose the I/O MMU to the hypervisor
  - Support SR-IOV PCI configuration

- **NICs**
  - NICs should be compliant with the SR-IOV and Sharing Specification.
  - NICs must support an e-Switch

- **OS**
  - PCIM implementation
  - Driver model/API

The implementation of SR-IOV requires the following PCIe capabilities:

- Alternative Routing-ID Interpretation (ARI)
- Function Level Reset (FLR)
- Address Translation Services (ATS)

**Alternative Routing-ID Interpretation (ARI)**

**PCIE ECN**

The PCI specifies the route ID as Bus (8-bits): Device (5 bits). Function (3 bits). However, on PCIe links, there is only one device - Dev 0. Therefore, the 5 bits can be combined with the Function bits. This is called ARI. It enables support for >8 functions, supporting up to 255 functions per device per bus. I/O PCIe devices, switch ports, and bridges upstream of ARI-compliant switches and endpoint devices must also implement ARI.
Function Level Reset PCIe ECN

The Function Level Reset (FLR) mechanism enables software to quiesce and reset hardware with Function-level granularity. FLR applies on a per Function basis. Only the targeted Function is affected by the FLR operation. FLR enables device reset at a per VF level.

Address Translation Services (ATS)

Memory Translation technologies such as those in Intel VT-x and VT-d provide hardware assisted techniques to allow direct DMA transfers. ATS allows device driver in a VM to transfer data directly to/from a device via DMA.

SR-IOV End Point

The SR-IOV device implements one or more physical functions with unique memory space, work queues, interrupts, and command processing for each virtual function. The device implements a Virtual Embedded Bridge (VEB) or Edge Virtual Bridge (EVB) in the hardware.

OS Support

The hypervisor must implement the PCI-Manager functionality to enable SR-IOV support. The VMM (i.e. hypervisor) configures an SR-IOV device to appear in the PCI configuration space as multiple functions. The VMM assigns one or more VFs (vNICs) to a VM. It maps the VF configuration space to the VM configuration space presented by the VMM.

The following OSes support SR-IOV:

- VMware ESX
- Microsoft Hyper-V
- Linux KVM

BIOS Support

The System BIOS must recognize SR-IOV devices so that enough MMIO space is allocated to encompass the requirements of VFs. System BIOS also enables ARI support to extend the number of PCIe functions from 8 to 255.

NIC Partitioning

It is typical for NICs to have one PCIe function per physical port. NIC Partitioning (NPar) mode enables the capability to divide a NIC physical port into multiple PCI physical functions with flexible line side transmit bandwidth capacity allocation. Partitions appear to the OS and network as separate NIC ports. A single Ethernet port appears in the PCI Configuration space as multiple physical function devices on the PCIe link. The PCIe link is unaffected.
Use Cases

NPar enables customers to replace multiple individual NICs with a single NPar-enabled NIC. This reduces switch port count and cabling complexity while maintaining network segmentation and isolation. In addition, flexible bandwidth allocation per partition allows for efficient use of the link.

An NPar-enabled NIC port can support the following server use cases:

- **Server Segmentation**: The partitions can be on separate subnets or VLANs. Embedded L2 switch for partition-to-partition communication.
- **Server High Availability**: The partitions will support NIC teaming including switch independent link failover and load balancing.
- **Physical and Virtual Server**: NIC Partitioning will be supported in both native OS and hypervisor based OS. In a Virtual Server, Emulated and Direct Assignment I/O of partitions to VMs will be supported.
- **Partitions can be made available for VM assignment (direct or emulated) and for application segmentation via VLAN or IP subnets.**

NIC Partitioning Benefits

The following are the key benefits of using NPar:

- **Multiple 1GE NICs can be replaced by a single 10GE or greater NIC**
  - Reduces number of ports and cables
  - Flexible transmit bandwidth allocation on the server Ethernet link
  - Efficient use of high speed physical links
  - I/O slot conservation through port consolidation
- **Allows a NIC to be partitioned into multiple functions with no OS or BIOS changes required and compatible with standard L2 switches. The solution is handled entirely by the NIC hardware and software.**
- **Allows transmit bandwidth allocation and rate control at a partition level on the server end.**
- **Allows partitions to assume various roles such as:**
  - iSCSI Offload
  - FCoE Offload
  - iWARP
  - RoCE
NPar vs. SR-IOV

NPar is limited to eight functions per device by the traditional PCI BDF routing ID interpretation. However, like SR-IOV, NPar can go beyond the eight function limitation by using PCI Express 3.0 ARI.

SR-IOV requires support from the operating system and BIOS for ARI while NPar works with current operating systems and BIOS software.

Another difference is that with NPar, the MAC addresses are pre-OS hardware assigned while with SR-IOV, they are locally administered MAC addresses. In SR-IOV, the VF usually (if not always) has a different device driver with different limitations than PFs. Until Linux added trusted VFs, a guest could not set the MAC address on the VF or put it in promiscuous mode. Table 14 lists the comparison between NPar and SR-IOV as they were perceived as of the writing of this book.
Table 14. NPar vs. SR-IOV

<table>
<thead>
<tr>
<th>Features</th>
<th>NPar</th>
<th>SR-IOV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypervisor Support Needed</td>
<td>No, Hypervisor agnostic - Feature limited (Static Direct Assignment, NIC Teaming)</td>
<td>Yes, support planned in RHEL5.4 SLES 11 Sp1, XenServer 5.6, VMware ESX 5.1, Windows Server 2012, and Windows Server 2016</td>
</tr>
<tr>
<td>Max # of partitions per device</td>
<td>8 (for non-ARI) &lt;br&gt; &gt;8 (when ARI is supported)</td>
<td>256</td>
</tr>
<tr>
<td>Monolithic Native OS Support</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Partitioning Control</td>
<td>NIC Firmware</td>
<td>Hypervisor</td>
</tr>
<tr>
<td>External Switch Requirements</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Target Environment</td>
<td>Physical (but also works in Virtualized OS)</td>
<td>Virtualized OS</td>
</tr>
</tbody>
</table>

On the host OS side, NPar will present up to eight PCI functions per device in the standard PCI configuration space when using traditional BDF routing ID interpretation or up to 255 PCI functions when using ARI. The actual number of functions available on a given NPar device is implementation-specific. Each function or partition will be assigned a unique MAC Address.

Figure 58. NIC Partitioning Example
Network Function Virtualization

Network Function Virtualization (NFV) is a new architecture that was developed to transition Telco proprietary hardware equipment to virtual appliances running on standard x86 servers. These network functions are instantiated by one or many Virtual Network Functions (VNFs).

Virtualization means that the network function or Virtual Network Function (VNF) is implemented in software and executed in a virtual machine, container, or other compute segmentation architecture. The VNFs provide switching, load balancing, controller, firewalling, or other similar functions.

NFV implementations may be single or coordinated with many VNFs chained together. VNF service chaining provides for multi-layer decision points and services at each NFV node. This allows processing of a packet through multiple functions such as switching and load balancing via the VNFs.

An NFV-ready NIC includes offloads and accelerations such as DPDK (also known as Packet Direct in Windows), Linux OVS offloads, and virtualization support such as SR-IOV to improve VNF performance.
iSCSI stands for Internet SCSI protocol that defines a mapping of the SCSI protocol over the TCP/IP protocol. It was originally defined in IETF RFC 3720 and more recently by RFC 7143. It is a block I/O storage access protocol. It uses SCSI Command Descriptor Blocks (CDBs) as well as networking standard protocols such as L2: Ethernet, L3: IP, and L4: TCP.

iSCSI can be implemented in software or hardware (known as iSOE covered later in the book)

<table>
<thead>
<tr>
<th>Application</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI</td>
<td>iSCSI</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>FCP</td>
</tr>
<tr>
<td>TCP</td>
<td>FCP</td>
</tr>
<tr>
<td>IP</td>
<td>FCP</td>
</tr>
</tbody>
</table>

**Figure 59.** Protocol Stacks

The protocol specifies two roles: Initiator and Target. The initiator interfaces with applications requiring remote disk access, sends commands/data to an iSCSI target, and receives responses/data from the iSCSI target. The target is the storage device or appliance where the disks reside. It allows for logical unit (LU) creation and logical unit number (LUN) masking via a storage controller and receives commands and sends responses from/to the initiator. A target can contain multiple LUs. Each LU has an address within a target called a LUN.

iSCSI Initiators are the SCSI devices that start the I/O process and iSCSI Targets are the iSCSI devices that respond to a request to perform an I/O process - read or write.

iSCSI in the server is implemented as a software initiator or a hardware initiator similar to the FC HBA offload model using an iSOE.
Protocol Stack

The iSCSI protocol can be implemented in the software as shown in Figure 61. It can use regular NIC with no offloads or it can use NICs with hardware offloads including stateless offloads and stateful offloads such as TOE and iWARP.

Figure 61. iSCSI Software Stacks
Connection

iSCSI uses a TCP connection with 3260 as the well-known port number. Communication between initiator and target occurs over one or more TCP connections. The TCP connections carry control messages, SCSI commands, parameters, and data within iSCSI Protocol Data Units (iSCSI PDUs). It includes single or multiple connections per session.

Session

The group of TCP connections that link an initiator to a target is an iSCSI session (loosely equivalent to a SCSI IT Nexus). TCP connections can be added and removed from a session. The initiator only sees one target across all connections within a session. An iSCSI session is established between an initiator and the storage target. There could be multiple initiators accessing a target—each with its own iSCSI session. Each session can be built over one or more TCP connections.

For iSCSI sessions with pending SCSI tasks, the adapter stops accepting new SCSI commands when the request for session termination is received and attempts to complete all the pending tasks. If tasks are not completed within a configurable timeout, the adapter aborts the pending tasks and closes the session.

![Figure 62. iSCSI Communications](image)

There are two session/connection phases: Login phase and Full Feature phase. The following functions occur between the initiator and target:

Login phase:
- Discovers targets via iSNS or SendTargets
- Creates a TCP connection
- Authenticates each party
- Negotiates operational parameters
- Creates or marks connection to a session
Data transfer is performed during the Full Feature phase.

In addition, there are different session types:

- **Normal** – A session used to transfer data between the initiator and a target.
- **Discovery** – A session used only for target discovery. The initiator sends a text command with a SendTargets key.

**Protocol Data Unit**

The initiator and target divide their communications into messages. The term iSCSI protocol data unit (iSCSI PDU) is used for these messages.

**iSCSI HBA**

The iSCSI HBA provides a full offload of the data path. The HBAs support the following standards:

- RFC 3720 (iSCSI)
- RFC 3721 (Naming & Discovery)
- RFC 3722, RFC3783 (Command ordering)
- RFC 3347 (Requirements and Design Considerations) RFC 4173 (Bootstrapping)
- RFC 4544 (Definitions of Managed Objects (MIB))
- RFC 1994 CHAP
FC initiators contain a number of Node Ports (N_Port) that connect to the Fabric Ports (F_Port) on switches. FC switches talk to each other using Expansion Ports (E_Port) before finally communicating with the N_Port on the storage array. This allows them to route traffic through the SAN to avoid data loss and congestion.

FCoE SANs adopt a virtual version of this configuration with a VN_Port talking to a VF_Port, and (if they support it) the network switches using VE_Ports to exchange data over an inter-switch link (ISL).

One major difference between an FC fabric and Ethernet network is intelligence. The FC fabric itself actively participates in access and routing decisions. The FC fabric has some intelligence (built-in services) and thus each FC switch makes decisions about where to send data traffic. Therefore, each stream of initiator-to-target traffic gets its own route through the SAN rather than sharing a single route as in an Ethernet LAN with spanning tree path management.

FCoE has the following benefits:

- I/O Consolidation - reduction of cables and network ports
- Stateless - simple protocol to implement
- Reuse of FC SAN tools and manageability
- No fragmentation and re-assembly
- No gateway for SAN connectivity
FCoE Protocol

FCoE is being defined in the FC-BB-5 standard. FCoE is a protocol for encapsulating and transporting FC frames over Ethernet networks. It is composed of FCoE Initialization Protocol (FIP) and Encapsulation operations. The entire original FC frame is encapsulated into an Ethernet frame - one FC frame to one EN frame.

<table>
<thead>
<tr>
<th>Application</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI</td>
<td>SCSI</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>Encapsulation</td>
</tr>
<tr>
<td>iSCSI</td>
<td>FCP</td>
</tr>
<tr>
<td>TCP</td>
<td>FCIP</td>
</tr>
<tr>
<td>IP</td>
<td>IP</td>
</tr>
<tr>
<td>Transport</td>
<td>Legacy Ethernet or DCB Ethernet</td>
</tr>
<tr>
<td></td>
<td>DCB Ethernet</td>
</tr>
<tr>
<td></td>
<td>Fibre Channel</td>
</tr>
<tr>
<td></td>
<td>Infiniband</td>
</tr>
</tbody>
</table>

Figure 63. Storage Protocol Stack

FCoEv1 does not use TCP/UDP/IP, therefore it is not routable. This limits the range of this technology to the data center.

FIP provides the following services: discovery of nodes and fabric login/fabric logout (FLOGI/FLOGO). Specifically, FIP defines the encapsulation of FLOGI, FDISC, FLOGO and ELP frames along with their reply frames.

Fibre Channel Forwarders (FCFs) use FIP to discover the nodes on the fabric and other FCFs. ENodes use FIP to log in with the fabric. Through FIP, the FCFs and ENodes on the fabric will establish virtual ports: VN_Port, VF_Port and VE_Port. The virtual ports coincide with the standard FC ports, N_Port, F_Port, and E_Port.

FCoE Addressing

There are two defined addressing methods: Server Provided MAC Address (SPMA) and Fabric Provided MAC Address (FPMA). If using SPMA, the MAC address provided by the server in the FLOGI request will be used. However, SPMA is not implemented by the industry.

FPMA uses the MAC address provided by the FCF in the FLOGI reply. Valid FPMA addresses will have the most significant 24 bits based on the FCF and the least significant 24 bits uniquely identifying the server ENode. FIP is used to assign/de-assign MAC addresses.

FCoE also defines several well-known MAC addresses used in FIP: all ENodes (ALL_ENODE_MACS) and all FCFs (ALL_FCF_MACS).
When a server ENode is connected to the fabric, it uses the following processes:

- FIP Discovery
- FLOGI
- FDISC
- FLOGO

The following are configured via the device driver on the adapter:

- FCF MAC address
- VN-Port MAC address
- Priority
- VLAN ID for the fabric that is connected to the adapter

**Discovery**

During FIP discovery, an ENode sends a Discovery Solicitation (DS) frame to all FCFs in the fabric. When an FCF receives a DS, it responds with a Discovery Advertisement (DA) to the ENode that sent the DS if it is configured to allow that ENode on the fabric.

When an FCF is added to the fabric, it sends a DS to all FCFs in the fabric. When an FCF receives a DS from another FCF, it sends a DA if it is configured to allow an Inter Switch Link (ISL) with the originating FCF. Additionally, FCFs periodically transmit DAs to all ENodes and FCFs as a heartbeat.

**Encapsulation**

FCoE encapsulation defines how all frames, other than the FIP frames, will be transmitted on the Ethernet network.
FCoE Frame

The FCoE frame encapsulates the entire FC frame including the FC CRC checksum. The FCoE MTU is 2.5 KB to accommodate the FC frame size of 2148 bytes plus the Ethernet and FCoE header. The FC payload inside the FC frame is 2112 bytes as shown in Figure 64:

- Encapsulates entire FC frame including CRC unchanged
- FCoE will require 2.5 KB to prevent fragmentation
  - Typical FC Frame is 2112 bytes

**Figure 64.** FCoE Frame
**FCoE Initialization Protocol**

This is the protocol that does the heavy lifting for FCoE. It encapsulates the FC link services and discovers the FCFs. FIP is used to log in and log out of the fabric.

- Encapsulates FC link services
- Discovery of FC entities
- ENodes find FCoE Forwarder (FCF) accepting VN Port connection
- FCF find FCF accepting VE Port connection
- Find out MAC address, FC_MAP, Switch Name, etc.
- Login/Logout of FCF
- Exchange link parameters

![FIP Header Diagram](image)

**Figure 65.** FIP Header

**Login**

An end-node goes through a login process when powered on to connect to the fabric as shown. It first discovers the FCF. Next, a fabric Login is requested and if successful, FC transactions can occur.

![Initialization Diagram](image)

**Figure 66.** Initialization
FCoE Adapter

FCoE adapters combine NIC functionality and FC functionality into one adapter. The OS sees a LAN device and a storage device. The FCoE function is hidden from the OS. It will load a storage device driver (i.e. Storport) for the FC function. A network device driver will be loaded for the LAN function.

When a packet is received, a filter is applied for the EtherType header field to identify if the packet is FCoE. The packet is then handled accordingly.

Offload Architecture

The FCoE offload adapter supports FCoE and FIP. Full offload of the data path for SCSI IO including offload for FC and FCoE is supported. Session initiation and teardown may be managed by host driver components, but once an FC session is established, all FC protocol involved in initiating and completing SCSI IOs is offloaded in the hardware.

Fibre Channel Protocol Offload

The adapter driver running in the host will provide SCSI command, destination target information, and DMA buffer addresses to the adapter. The adapter uses Fibre Channel Protocol (FCP) to issue command to the target, handle all the phases of the command execution (command, data, and status), and finally return the completion status to the host driver. The adapters transfer the data via DMA to/from the DMA buffers supplied by the host (Direct Data Placement) and provide the completion status to the host.
FCoE Attributes

FCoE relies on several parameters such as MAC Address formulation and Node Name derivation.

MAC Addresses

The offload adapter supports a Fabric Provided MAC Address (FPMA). An FCF switch will advertise a 24-bit FC-MAP value to the host via FIP support. The host will combine the FC-MAP with the FC-ID as its FCoE FPMA MAC.

The offload adapter assigns a burnt-in MAC address in NVRAM for the FIP.

Each Ethernet port supports the following MAC addresses:
- LAN
- iSOE
- FCoE-FIP

FC Node Name - WWPN, WWNN

The offload adapter provides the ability to configure its FC World Wide Node Name (WWNN) and FC World Wide Port Name (WWPN). World Wide Name (WWN) formats are specified in the FC-FS-3 specification section 15 (Name_Identifier_Formats). The WWPN and WWNN will be formed using the FCoE-FIP MAC address.

Node Name Persistence

The configured node name and port name for an adapter along with other related information are stored on the adapter in non-volatile storage.

Disconnecting Sessions

For FC sessions with pending SCSI tasks, the adapter stops accepting new SCSI commands when the request for session termination is received and attempts to complete all pending tasks. If tasks are not completed within a configurable timeout, the adapter aborts the pending tasks and closes the session.

Class 3 Service

The adapter supports class 3 services. Support for other classes of services is optional.

FCF Discovery using FIP

The FCoE adapter implements discovery of FCFs by using the FIP as specified in the FC-BB-5 standard.

FC target discovery

After discovering FCFs, the FCoE adapter discovers FC targets by performing PLOGI to F_Port and interacting with the FC name server, as specified in the FC-BB-5 standard.
**FCoE Topologies**

The following are possible deployment configurations for FCoE. Most deployments are single hop Top of Rack (ToR).

**Non-Converged Topology**

As shown in Figure 68, servers in the data center today connect to the LAN via GE links and to the SAN via FC links. This creates complexity in cabling for both GE and FC switches.

![Figure 68. Multiple I/O Fabric Connectivity](image_url)

**Server I/O Consolidation**

FCoE enables I/O convergence starting in the server by replacing the LAN and SAN host controllers with Converged Network Adapters that provide both LAN and SAN traffic support over 10GE. This reduces the number of switch ports and cabling complexity.

![Figure 69. Single FCF](image_url)
Multi-FCF, Single FCF Fabric-Multi-hop DCB

As data center bridging is deployed in the data center, FCoE can be moved to end of row when enabling multi-hop DCB implementations.

![Figure 70. DCB in the DC](image)

Native FCoE Targets

The final topology will be enabled by native FCoE targets where an FC switch Fabric is not required and the FC SAN is replaced or complemented by an FCoE SAN.

![Figure 71. FCoE Targets](image)
**Fibre Channel Backbone - 6 (FC-BB-6)**

FC-BB-6 standardizes the VN to VN (VN2VN) architecture and introduces the controlling Fibre Channel Forwarder (FCF)/FCoE Data Forwarder (FDF) topology. VN2VN is a way to directly connect FCoE end-nodes (Virtual N_Ports) without FC and FCoE switches (or FC Forwarders).

This concept is also sometimes known as Ethernet Only FCoE. The controlling FCF/FDF topology includes traditional FCFs and FCoE Data Forwarders (FDF). An FDF acts as a layer 2 switch, similar to a network access or edge device. The term FCoE aware bridge has also been used to describe an FDF.

The VN2VN proposal will allow FCoE to work in a standard DCB switching environment without the presence of an FCF. An FCF allows for bridging between servers which communicate with FCoE and storage devices which in turn communicate with the traditional FC.

As DCB switches and FCoE storage become more prevalent, the FC-BB-6 standard will allow for end-to-end FCoE connectivity in either a point-to-point (P2P) or DCB mesh environment. This will result in lower cost for FCoE environments.
DCB enables Ethernet fabrics to support lossless flows required during congestion conditions. The IEEE is developing the underlying standards for DCB under the IEEE 802.1 Working Group based on the Converged Enhanced Ethernet (CEE) Authors Group Version 0 initial draft. However, the IEEE and CEE specifications are different and not compatible. The pre-IEEE standard implementations based on the CEE Authors Group Version 0 are as follows:

- The Priority-based Flow Control (PFC) Version 0 Specification was submitted to the IEEE 802.1Qbb working group.
- The Enhanced Transmission Selection (ETS) Version 0 Specification was submitted to the IEEE 802.1Qaz working group.
- The Data Center Bridging eXchange (DCBX) Version 0 Specification was submitted to the IEEE 802.1Qaz working group.

DCB is a class of Ethernet capabilities that improve the quality of service of networks for data center applications.

Conventional Ethernet is a best-effort network topology and as such it drops packets in response to traffic congestion. Protocols such as FCoE do not have guaranteed delivery quality of service and require a lossless underlying physical transport. DCB adds enhancements to Ethernet to reduce packet loss due to congestion. DCB development is focused on FCoE, but it can also benefit iSCSI by improving handling of network congestion conditions offered by TCP.

DCB is an umbrella term used by the DCB Task Group within the IEEE 802.1 Working Group. The charter of the DCB TG is to provide enhancements to existing 802.1 bridge specifications to meet the requirements of protocols and applications in the data center.
Legacy Ethernet Frame Format

Legacy Ethernet allows for the use of a tag as shown in Figure 72. It includes a Priority Tag for the purpose of tagging frames with different priorities.

![Frame Format](image1)

IEEE 802.1p Packet Priority

The IEEE 802.1p tag is used by the NIC to indicate the priority to the link partner. This link, between the NIC and a switch, must be set up as a trunk. The Switch transmitting interface assigns priority, via the tag, to packets before forwarding. IEEE 802.1p recommends fixed priority as traffic class-queue association.

![Priority Tag](image2)
Legacy Flow Control

The base NIC implements a PAUSE frame based flow control. The flow control operation is limited to full duplex. The NIC driver configures the hardware via the following parameters:

- High water mark – RX packet buffer fullness level above which the hardware will transmit XOFF frame.
- Low water mark – RX packet buffer fullness level under which the hardware will transmit XON frame.
- Flow control timer value – Timer value to be inserted into the transmitted XOFF frames.

Legacy Ethernet Flow Control is shown in Figure 74. It has the following limitations:

- Legacy Flow Control affects the entire link.
- Legacy Flow Control is not required on both ends of the link.

The device will continue to support Legacy Flow Control in addition to PFC.

**Figure 74.** Flow Control
Priority-based Flow Control

Priority-based Flow Control (PFC), IEEE 802.1Qbb, is intended to eliminate frame loss due to congestion. This is achieved by a link level flow control mechanism similar to the IEEE 802.3x PAUSE. However, PFC operates on individual priorities, which are the eight traffic classes defined in IEEE 802.1Q and marked in a frame by using the 3-bit Priority Code Point field in the Virtual LAN (VLAN) tag. It indicates the frame priority level from 0 (lowest) to 7 (highest), which can be used to prioritize different classes of traffic (voice, video, data, etc.). PFC is strictly link-based in scope.

The goal of this mechanism is to ensure near zero loss due to congestion in DCB networks. The PFC standard specifies protocols, procedures, and managed objects that enable flow control per traffic class on IEEE 802 full-duplex links. PFC has the following characteristics:

- PFC allows link flow control to be performed on a per-priority basis.
- PFC is used to inhibit transmission of data frames associated with one or more priorities for a specified period of time.
- A VLAN unaware end station can use PFC by sending traffic as priority-tagged and by ignoring the VLAN ID in received frames.

![Priority-based Pause Diagram](image)

**Figure 75.** Priority-based Pause
Flow Control Frame

Figure 76 shows the legacy and DCB Pause frame format. The DCB PFC frame added the ability to set Pause for any of the eight priority queues. Ideally, there are eight hardware queues to match in the NIC controller.

![Figure 76. PFC Frame](image_url)

Enhanced Transmission Selection

ETS, IEEE 802.1Qaz, provides definitions and an operational model for priority processing and bandwidth allocation on converged links in end stations and bridges in a DCB environment. Using priority-based processing and bandwidth allocations, different traffic classes within different traffic types, such as LAN, SAN, IPC, and management can be configured to provide bandwidth allocation, low-latency, or best effort transmit characteristics for traffic that can handle packet discards. To this end, the ETS standard includes the following tasks:

- Specifies the DCB Capability Exchange Protocol (DCBX), a mechanism that allows discovery and configuration of DCB protocols.
- Defines use of traffic class to group priorities together for the purpose of ETS bandwidth allocation.
- Defines a set of bandwidth configuration parameters that are used to configure the percentage of bandwidth assigned to each traffic class.

ETS provides a common management framework for assignment of bandwidth to traffic classes.
This standard specifies enhancements to the packet scheduler in the NIC to support allocation of bandwidth among traffic classes. When the offered load in traffic class does not use its allocated bandwidth, ETS will allow other traffic classes to use the available bandwidth. The bandwidth-allocation priorities will coexist with strict priorities. It will include managed objects to support bandwidth allocation. As specified by the referenced PFC specifications, the IEEE ETS implementation will operate as follows:

- Allows for sharing of bandwidth between priorities carrying bursty high offered loads rather than servicing them with strict priority
- When traffic at a priority level does not use its allocation, ETS allows other priorities to use that bandwidth
- Includes implementation of DCBX protocol that controls the application of ETS and PFC configuration parameters

**Data Center Bridging Exchange Protocol**

Data Center Bridging Exchange Protocol (DCBX) is a protocol to exchange configuration information with directly connected peers. The protocol may also be used to detect misconfiguration and to configure the peer for the following:

- ETS
- PFC
- Application TLV
Table 15. DCB Protocol Features

<table>
<thead>
<tr>
<th>Features</th>
<th>Standard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority Flow Control (PFC)</td>
<td>IEEE 802.1Qbb</td>
<td>Enables multiple traffic types to share a common Ethernet physical link without interfering with each other PFC allows link flow control to be performed on a per-priority basis</td>
</tr>
<tr>
<td>Enhanced Transmission Selection (ETS)</td>
<td>IEEE 802.1Qaz</td>
<td>Enables transmission bandwidth management / rate limiting per traffic class</td>
</tr>
<tr>
<td>Congestion Notification (CN)</td>
<td>IEEE 802.1Qau</td>
<td>End-to-end congestion management of long-lived data flows</td>
</tr>
<tr>
<td>Data Center Bridging Capability Exchange Protocol (DCBCX)</td>
<td>IEEE 802.1Qaz</td>
<td>LLDP-based protocol to exchange link configuration parameters for PFC, ETS, etc.</td>
</tr>
<tr>
<td>MAC Control Frame for Priority-based Flow Control</td>
<td>IEEE 802.3Qbd</td>
<td>Defines a MAC Control Frame to support 802.1Qbb Priority-based Flow Control</td>
</tr>
</tbody>
</table>

DCBX uses LLDP (see IEEE 802.1AB) to exchange attributes between two link peers enabled for DCB.

DCB exchanged attributes are packaged into Organizationally Specific Type, Length, Values (TLVs). The OUI used for the DCBX is the IEEE 802.1 OUI (i.e., 0x0080c2).

The NIC device implements the LLDP state machines and TLVs for capability/attribute exchange as defined by the IEEE ETS/DCBX standard. Management software will be needed to examine the exchanged parameters to detect any misconfiguration and take action to resolve it.

A Willing port will set its operational attribute to that indicated in the received TLV if the received TLV has the Willing bit set in the DCBX header. If both the local port and remote port are Willing, then the attribute values of the port with the lower numerical MAC address will take precedence.

The NIC driver may wish to delay the use of the link (hold presenting Link Up to the OS stack) while the DCBX state machine is in the process of passing and possibly setting attributes.
ETS TLVs
The IEEE ETS standard defines the following TLVs to be exchanged via DCBX:

- ETS Configuration TLV
  - Asymmetric
  - Willing
  - Max TC
  - Priority Assignment Table (Priority to TC)
  - Traffic Class BW Assignment
  - Transmission Selection Algorithm

PFC TLVs
The IEEE PFC standard defines the following TLVs to be exchanged via DCBX:

- PFC Configuration TLV
  - Symmetric
  - Willing
  - PFC Capability (# of TC that support PFC)

Application Priority TLV
The IEEE defines the following TLVs to be exchanged via DCBX:

- Symmetric
- Informational
- Application Priority Table (Protocol ID)

The device will implement an Application TLV for iSCSI based on the protocol port number.
There are workloads and applications that require low latency. A typical NIC has a back-to-back, one way latency of 10-12 μs approximately. Some applications require much lower latency in the order of nanoseconds. Traditionally, low latency has been a focus for High Performance Computing Clusters and High Frequency Trading. However, new applications are taking advantage of low latency technologies. Typically, OS networking stacks are not optimized for best performance, but rather focus on reliability. Low latency requires bypassing this software stack and implementation of zero copy and context switching avoidance.

**Figure 77.** Typical Network Stack

- TCP IP handled by OS Kernel
- Simple, inexpensive hardware
- Context switching and interrupt overhead
- Cache thrashing and inefficient data handling
- High CPU utilization over multiple GE links and 10GE
- Limited throughput over multiple GE links for receive traffic
- Excessive application latency
Key Factors
The following are the key aspects for reducing the NIC latency:

- Kernel bypass
- Zero copy
- Context switching avoidance

Applications
Some of the applications that require low latency include the following:

- SMB Direct
- Live VM Migration
- Non-Volatile Memory over PCIe (NVMe) over Fabrics
- Network Function Virtualization (NFV)
- High Frequency Trading (HFT)
- High Performance Computing (HPC)

SMB Direct
SMB is a client/server protocol for file/print sharing in Windows. SMB Direct added RDMA transport support based on Network Direct Kernel Programming Interface (NDKPI). It defines send, write, and read operations over RDMA. It supports multiple RDMA channels and reduces CPU utilization and latency to near local levels for file I/O. SMB Direct also supports a transparent failover mechanism between RDMA and TCP/IP. Initial connection is over TCP/IP and it is used to discover if there is an RDMA device. If there is RDMA, it switches over to an RDMA connection. NDKPI supports IB, iWARP, and RoCE.
Low Latency Protocols

There are several protocol approaches available to improve latency of the NIC such as the following:

- DPDK
- RDMA
  - iWARP
  - RoCEv1
  - RoCEv2
- OpenOnload

In addition, SR-IOV lowers the NIC latency by bypassing the hypervisor vSwitch on the data plane. The VMs have direct access to the SR-IOV VFs instead of connecting to the kernel vSwitch in the software track.

Data Plane Development Kit (DPDK)/Packet Direct

DPDK (also known as Packet Direct in Windows) is a set of Data Plane Libraries and Optimized NIC Drivers in Linux User Space. It provides Queue and Buffer Management, Packet Flow Classification, and support for Poll-Mode NIC Drivers. It implements a run to completion model or pipeline model where all devices are accessed via polling.
DPDK Libraries

DPDK defines a set of libraries for user space application development. Some of the key libraries are described below.

Memory Manager

Responsible for allocating pools of objects in memory. A pool is created in a huge page memory space and uses a ring to store free objects. It also provides an alignment helper to ensure that objects are padded to spread them equally on all DRAM channels.

Buffer Manager

Significantly reduces the time the operating system spends allocating and de-allocating buffers. DPDK pre-allocates fixed size buffers that are stored in memory pools.

Queue Manager

Implements safe lockless queues, instead of using spinlocks that allow different software components to process packets, while avoiding unnecessary wait times.

Flow Classification

Provides an efficient mechanism, which incorporates Intel Streaming SIMD Extensions (SSE) to produce a hash based on tuple information so that packets may be placed into flows quickly for processing, thus greatly improving throughput.

PacketDirect Provider Interface

The PacketDirect Provider Interface (PDPI) extends NDIS with an accelerated I/O model for both physical and virtual environments. This can increase the number of packets processed per second by an order of magnitude and significantly decrease jitter when compared to the traditional NDIS I/O path.

The PDPI allows NIC drivers to expose their high-performance send and receive functionality to the Windows OS. The functions implemented are a subset of the complete MiniPort driver and are generic to all NICs that implement Packet Direct.

RDMA

RDMA has been implemented and deployed in the form of IB for around 10 years. It has carved a place in the HPC vertical as a high throughput and low latency fabric. In addition, iWARP was standardized in 2003 to improve application performance via kernel bypass and zero copy.

Adoption has been slow historically; however, recently there has been a resurgence of interest for RDMA for storage use cases such as SMB Direct and NVMe over Fabrics.
The data flow on a standard NIC operating on the OS network software stack causes a bottleneck impacting throughput, CPU utilization, and latency especially when dealing with high performance computing environments or applications. RDMA is required to eliminate multiple memory copies that take place during the data transfer. The memory speeds have not kept up with the current increased rate of CPUs and interconnects. This means the CPU is involved in data processing and has to wait for the data in memory.

RDMA technologies enable a more efficient way of data transfer by implementing zero copy and OS stack bypass to remove the CPU in the data path. This reduces latency and increases the throughput by supporting DDP from one node to another over a network.

Typical NIC flow:

1. Application Data is buffered in host memory
2. Data is copied to socket layer buffers in host memory using CPU
3. Data is copied to transport layer buffers in host memory using CPU
4. Data is copied to NIC driver layer buffers in host memory using CPU
5. Data is copied to the NIC FIFO packet buffers using DMA

RDMA eliminates the data copies that require intervention by the host CPU.

---

**Figure 79.** Buffer Copies
RoCE stands for RDMA over a Converged Ethernet network. RoCE benefits from DCB to enable a near lossless operation on the network. DCB includes PFC (Priority-based Flow Control), ETS, and DCBX (Data Center Bridging eXchange). This is helpful when running RoCE traffic plus storage (iSCSI, FCoE) traffic over the same interface.

Figure 80. RDMA Concept

Figure 81. IB vs. RoCE
RoCE v1 and v2

RoCE v1 specifies the encapsulation of IB into Ethernet frames while maintaining the IB transport. Therefore, v1 is not routable and limited to one VLAN. The RoCEv1 frames are identified via Ethtype.

RoCEv2 or Routable RoCE (RRoCE) has IPv4 and IPv6 support to allow routability across different subnets. The RoCEv2 frames are identified via UDP port number. The transport layer for RoCEv1 was changed from IB to UDP/IP in RoCEv2.

![RoCE Headers](image)

Figure 82. RoCE Headers

iWARP

iWARP is an Internet Engineering Task Force (IETF RFC 5040) update of the RDMA Consortium’s RDMA over TCP standard. This standard provides for zero-copy transmission over legacy TCP. iWARP runs RDMA, MPA, and DDP over TCP/IP.

iWARP is a protocol that reduces latency on applications written to the RDMA API. Examples of an RDMA API are Linux OFED, Windows NDKPI, and Windows SDP. The RDMA Protocol Verbs Specification specifies the RDMA NIC (RNIC) hardware, software, and firmware. The RNIC interface defines semantics of the RDMA services and can be implemented through a combination of hardware, software, and firmware.

A verb is an operation performed by the RNIC interface. A verb consumer is a software that uses the RNIC to communicate with other nodes. The verbs manage connection state, memory, and queue access. They submit work to RNIC and retrieve work and events from RNIC. The RNIC interface converts Work Requests to Work Queue Elements. It also converts Completion Queue Elements to Work Completion.
**Figure 83.** iWARP Stack

- SQ - Send Queue
- RQ - Receive Queue
- SRQ - Shared RQ
- QP - Queue Pair
- QP - SQ + RQ
- CQ - Completion Queue

**Figure 84.** iWARP NIC
**RDMA Messages**

A Remote Direct Memory Access Write (RDMA) uses an RDMA Write Message to transfer data from the Data Source to a previously advertised buffer at the Data Sink (or destination).

The Remote Peer, which in this case is the Data Sink, enables the Data destination buffer space to determine where to write the data.

A Remote Direct Memory Access Read transfers data to a Tagged Buffer at the Local Peer, which in this case is the Data Sink, from a Tagged Buffer at the Remote Peer, which in this case is the Data Source.

The RDMA Read Request Message uses the DDP Untagged Buffer Model to deliver the Steering Tag, starting Tagged Offset and length for both the Data Source and Data Sink Tagged Buffers to the remote peer’s RDMA Read Request Queue.

**RDMAP**

The RDMAP layer, as specified by RFC 7306 from the IETF, provides read and write services directly to applications and enables data to be transferred directly into upper-layer protocol (ULP) buffers without intermediate data copies.

**DDP**

The RDMA Read Response Message uses the Direct Data Placement (DDP) Tagged buffer model to deliver the data source’s Tagged buffer to the data sink, without any involvement from the ULP at the data source.

DDP provides a buffer model that enables the Local Peer to advertise a named buffer (Tagged buffer). A reliable, in-order delivery semantics is specified for both Tagged and Untagged buffer models. DDP provides for segmentation and reassembly of ULP messages.

The Tagged buffer data transfer model implements a mechanism to send the data source an identifier for the ULP buffer. This tag is referred to as a Steering Tag (STag). The data source ULP now has an STag for a destination ULP buffer. The data source can then request DDP to send the ULP data to the destination ULP buffer by specifying the STag to DDP.

On the other hand, the Untagged buffer data transfer model supports the data transfer to occur without requiring the data sink to advertise a ULP Buffer to the data source.
MPA

Marker PDU Aligned Framing (MPA) is designed to work as an adaptation layer between TCP and the DDP protocol as described in RFC 5041. It preserves the reliable, in-order delivery of TCP, while preserving higher-level protocol record boundaries that DDP requires. MPA is fully compliant with the applicable TCP RFCs and can be utilized with the existing TCP implementations. MPA also supports integrated implementations that combine TCP, MPA, and DDP to reduce buffering requirements in the implementation and improve performance at the system level.

![iWARP Header](image1)

**Figure 85.** iWARP Header

![iWARP Block Diagram](image2)

**Figure 86.** iWARP Block Diagram
iWARP provides the semantics to enable Remote Direct Memory Access between peers. It identifies buffers and controls the transfer of data between ULP peers. iWARP provides completion notifications to the ULP. RDMAP provides read and write services directly to applications. It enables data to be transferred directly into ULP Buffers without intermediate data copies. It enables a kernel bypass implementation.

**OpenOnload**

OpenOnload is a high performance network stack from Solarflare that reduces latency and CPU utilization and increases message rate and bandwidth. OpenOnload runs on Linux and supports TCP/UDP/IP network protocols with the standard BSD sockets API and requires no modifications to applications. It achieves performance improvements in part by performing network processing at user-level, bypassing the OS kernel entirely on the data path. Networking performance is improved without sacrificing the security and multiplexing functions that the OS kernel normally provides.

OpenOnload comprises a user-level shared library that intercepts network-related system calls and implements the protocol stack and supporting kernel modules. It is compatible with the full system call API, including those aspects that are usually problematic for user-level networking, such as fork(), exec(), passing sockets through UNIX domain sockets, and advancing the protocol when the application is not scheduled.
There are two widely used security protocols for data in flight over a network: IPSec and SSL/TLS. This book focuses on IPSec and only includes SSL to highlight key differences.

IPSec is a suite of protocols that provides secured communications over the Internet at the network layer. IPSec provides data source authentication, data integrity, confidentiality, and protection against replay attacks. This book provides a general overview of IPSec and its various components.

Secured communications over public networks and between private networks have long been an issue of great concern. The continued growth of e-commerce over the Internet, the demand for outsourcing of Web hosting services, the cloud, and the increase of Application Service Providers (ASPs) have heightened the need for ubiquitous security.

Today, most network traffic for both the Internet and corporate intranets is based on TCP/IP. However, since the original Internet Protocol (IP) failed to define any structures for security, application layer implementations, such as Secure Sockets Layer (SSL) and Secure HyperText Transfer Protocol (S-HTTP) have been used to provide data security over the Internet.

**Figure 87.** SSL
SSL creates a secure connection between a client and a server over which any amount of encrypted data can be sent. Web pages that require an SSL connection start their URLs with https instead of the normal http. However, these implementations require that both the sending and receiving stations run the required application software or Web browser. In addition, only the data to and from the Web server is secured. SSL runs above TCP/IP and below high-level applications, such as Lightweight Directory Access Protocol (LDAP) and Internet Messaging Access Protocol (IMAP).

Internet Protocol Security

IPSec is an Internet Engineering Task Force (IETF) standard that provides security at the network layer for data in-flight. This allows for more flexibility during its implementation. IPSec allows for private and secure communications over the public Internet regardless of the application or higher level protocols. Other IPSec characteristics include authenticating both senders and receivers, making data confidential via encryption, assuring data integrity, and working with any IP-based application.

IPSec can protect confidential data, such as human resources data, medical information, payroll records, and any other sensitive information, that is transferred within a local network (intranet) or across the Internet by limiting data access to only authorized users.

IPSec Components

The IPSec cryptography-based security technology suite defines the following header extensions to IPv4:

- Encapsulating Security Payload (ESP) header (IP Protocol 50)
- Authentication header (AH) (IP Protocol 51)

The other major component of the IPSec protocol suite is the Internet Security Association and Key Management Protocol (ISAKMP), which is used to implement the Internet Key Exchange (IKE) protocol. The ISAKMP method securely authenticates and establishes security associations (SAs).

The two basic modes for implementing the header extensions in an IP transaction include:

- Transport mode: Supports client-to-client or client-to-server communications with no intervening security gateways
- Tunnel mode: Supports remote access and site-to-site secured communications
The ESP Header

As defined in Request for Comments (RFC) 2406, the ESP header provides data encryption, data origin authentication, anti-replay, and data integrity services for IP packets. ESP operates at the network or transport layer; for example, ESP can be used to secure an FTP session by encrypting all data transmitted during the session.

The ESP header also contains a Security Parameters Index (SPI) field that is used as an index to identify the appropriate SA to use when processing an IPSec packet. The SPI is an arbitrary number established by the destination host using IKE. The SPI is authenticated but not encrypted, because this field needs to be in clear text (normal text) for the destination host to identify the encryption algorithm and key used by the SA Initialization Vector (IV).

An ESP SA defines algorithms for encryption and data authentication. Encryption is a mathematical operation that transforms normal text (clear text) into cipher text, which appears as a series of random characters. The cipher text is a function of a key and the clear text data.

ESP provides protection from replay attacks by providing a sequence number within the header. The sequence number, a unique value inserted into the header by the sender, is used to determine whether the packet is a duplicate that should be dropped. When the sequence number reaches a predefined maximum, a new SA is established to restart the sequence number from zero.

Figure 88. IPSec ESP Packet

Figure 89. 3DES Process
Data Encryption Standard (DES) and 3DES are symmetrical encryption algorithms. They use the same secret key to encrypt and decrypt the data. As shown, 3DES uses three different 56-bit keys to produce the cipher text. It decrypts the encrypted result from key X by using a different key. Finally, it encrypts the result of key Y by using key Z.

ESP encrypts the payload by using cipher algorithms, such as DES and 3DES, which are based on a 56-bit key and a 168-bit key, respectively. Cipher block chaining (CBC) is used with both DES and 3DES; the amount of data to be encrypted must be a multiple of the block size of the cipher. Hence, data is padded to achieve this result. CBC utilizes an IV to ensure that identical blocks of clear text will not result in the same cipher text.

The ESP trailer contains the necessary padding, length of the pad, next protocol after ESP, and authentication data or digest. The authentication data field, which is used to validate the authenticity of the packet, is the digest generated from a keyed hash function used for data integrity. ESP uses Hashed Message Authentication Code-Message Digest 5 (HMAC-MD5) or Hashed Message Authentication Code-Secure Hash Algorithm (HMAC-SHA) as authenticator algorithms. The output of these operations is a message digest used to verify the authentication of the data.

In general, the process for the received ESP packet is to verify the sequence number, verify integrity of the data (authenticate), and decrypt the data.

**AH Protocol**

An AH packet contains an AH between the IP and TCP headers. The AH, defined in RFC 2402, provides security services such as data integrity, data source authentication, and protection against replay attacks.
Figure 91. Authentication Header

AH provides data and address integrity without encryption. It differs from ESP in that, it not only does not provide encryption, but also no trailer exists at the end of the packet. The message digest, created by the authentication algorithms, uniquely identifies each IP packet based on a secret key. The AH, shown in Figure 91, contains the following fields:

- SPI - helps locate the SA to process the packet
- Sequence number - resists replay attacks by rejecting duplicate packets
- Authentication data - validates the source of the data

Figure 92. Hashing

HMACs use a single key for generating and verifying the authentication information, as shown in Figure 92. Hash functions take a variable-sized message (data) as input, compress it, and produce a fixed-size digest. The resulting digest is attached to the header. Verification at the destination entails hashing the shared secret key with the data and comparing the result with the digest in the header.

AH uses HMAC over the IP datagram to create the authentication data. The receiver verifies the integrity of the data after packet reassembly. Since AH authenticates the
outer IP header, it must be aware of IP header fields that change when processed by routers. These fields are omitted from the authentication calculations.

When both AH and ESP are protecting the same data, the AH is always inserted after the ESP header. The AH is simpler than the ESP header because it does not provide for confidentiality. The AH has no trailer because there is no need for padding and a pad-length indicator. There is also no need for an IV.

The authentication data field is a variable-length field that contains the result of the integrity-checking function. AH implements mandatory authenticators, such as HMAC-SHA-256+.

AH, like ESP, can be used in either the transport mode or tunnel mode. The difference is in the data being protected. In the transport mode, the upper-layer protocol is protected while in the tunnel mode, the entire IP datagram is protected. The SHA hashing algorithm adds 20 bytes to each packet while MD5 adds 16 bytes. SHA is considered a slightly more secure but slower executing algorithm.

**Authenticating IPSec**

IPSec computers must verify each other (authenticate) before they can begin secured communications. IKE performs authentication during the initial negotiation phase. Authentication can be implemented by using one of the following methods:

**Pre-shared key**

The IPSec computers exchange a previously shared password to verify each other. This secret string must be communicated using an out-of-band mechanism, such as through telephone, face-to-face, or other direct communications. This exchange should not be done over the same unsecured channels that IPSec is trying to secure.

**Certificate Authority**

IPSec computers use a common Certificate Authority (CA) to verify the identity of each other. Each computer registers with the CA, such as VeriSign or Entrust. The CA then authenticates each computer before they can engage in trusted communications.

**Kerberos Version 5 Protocol**

The database must have an SA before IPSec can secure an IP packet. The SA can be created manually or dynamically using IKE. The purpose of IKE is to negotiate an IPSec SA and populate the SA database.

**Internet Key Exchange**

As described in RFC 2409, IPSec uses IKE to create shared security parameters and authenticated keys-SAs-between the IPSec cryptographic end points. Two IPSec peers use IKE to establish a shared and an authenticated key.
ISAKMP defines the operation and language constructs used by IKE. OAKLEY is the key determination protocol used by IKE to authenticate the Diffie-Hellman exponent exchange. Therefore, IKE is a hybrid protocol that defines a way of deriving authenticated keying material and negotiating shared security policy based on ISAKMP and OAKLEY.

ISAKMP defines packet formats, the retransmission timer, and the programming language. IKE uses the ISAKMP common framework and procedures for the creation and management of SAs. A Domain of Interpretation (DOI), covered in RFC 2407, is used to document how IKE negotiates IPSec SAs.

IKE automates key exchange to deliver keys safely based on a Diffie-Hellman key-exchange protocol. Diffie-Hellman is a one-way function to securely exchange a shared secret over an untrusted communications channel. It is based on the exponentiation of prime numbers. A public value is exchanged and exponentiated to create the secret value. Diffie-Hellman assumes that the hosts, via passwords (pre-shared keys) or digital certificates, know the identities of the two IPSec end points. Once the hosts authenticate the end points, IKE exchanges information to establish a shared key. The hosts share configuration information by exchanging common SPIs that specify the encryption algorithm, authentication algorithm, and security keys to use when establishing an SA.

IKE SAs (different from IPSec SAs) define algorithms to encrypt IKE traffic and define how to authenticate the IPSec end points. IKE uses the two phases of ISAKMP. The first phase establishes the IKE SA and the second phase uses that SA to negotiate SAs for IPSec.

Unlike the IPSec SA, the IKE SA is bidirectional. IKE is a request-response protocol, where one party is the initiator and the other is the responder. Once the IKE SA is established, it may be used to protect both inbound and outbound traffic. The IKE SA has various parameters that are negotiated between the two IPSec end points. These parameters are referred to as the protection suites, and they include the following:

- Encryption algorithm
- Hash algorithm
- Authentication method
- Diffie-Hellman group

The protection suites are negotiated between the peers as part of the first messages they exchange. Each side maintains some secret information that, once authenticated, is used to protect IKE messages and derive keys for other security services.

The keys used for the IPSec SA are derived from the IKE shared key. In addition to the IKE key, other parameters, such as the Diffie-Hellman group number, nonces (random numbers), and security association parameters are used to ensure perfect forward secrecy.
IPSec Security Associations

The IPSec SA is the contract between two communicating entities; it specifies the IPSec protocols used for securing the packets. The SA provides an association among the security services, active key, data to be protected, and the end points.

![Security Association Diagram](image)

**Figure 93.** Security Association

The SA is a one-way simplex operation. SAs are established for processing outbound and inbound packets. An SA exists for each IPSec protocol. If both AH and ESP are being used, then a separate SA is established for each protocol. The SA parameters are stored in the Security Association Database (SADB). SAs can be created manually or dynamically. A manual SA has no lifetime limits and must be deleted manually, or it will continue to be active indefinitely. Dynamic SAs have a lifetime that is negotiated by the key management protocol.

This is an important factor, because to ensure security, keys must be refreshed regularly. The creation and deletion of SAs are the two most important tasks handled by the SA management protocol, such as IKE. The SA management protocol requires an interface between the user applications and the operating system kernel to manage the SADB. The SA creation is a two-step process: The SA management protocol negotiates the parameters of the SA, then it updates the SADB with the SA.

The SAs can be deleted manually or by using IKE. IPSec does not refresh keys; the existing SA must be deleted and a new one must be created. A new SA is negotiated before the existing SA expires. Typical reasons for an SA to be deleted is that the lifetime has expired, the keys are compromised, or the number of bytes encrypted/decrypted or authenticated using this SA has exceeded a threshold set by the security policy.

The SPI is used to determine a pointer to the specific SA in the database that will be used when applying the security policy. The destination selects the SA to process a packet via the SPI in the IPSec header. On the other hand, the source identifies the SA to secure the packet via the security policy selectors. Once the SA is created and added to the SADB, secure packets start flowing.
**IPSec Security Policies**

An IPSec security policy defines the action to be applied to a packet. The policy is stored in a database called the Security Policy Database (SPD), which is indexed by selectors. Security policies allow for different levels of security to be applied to traffic on the same network. A security gateway may require that all traffic between a protected subnet and another subnet be encrypted with DES and authenticated with HMAC-MD5, while 3DES and HMAC-SHA would be applied to Telnet traffic to a mail server from a remote subnet.

Policy management is required to add, delete, and modify policy. The SPD is stored in the operating system kernel. The SPD defines the traffic to be protected, how to protect it, and who shares in the protection. It must be consulted for each packet entering or leaving the IP stack. Once consulted, the SPD defines three actions based upon a traffic match, as identified by the following selectors:

- **Discard**: Do not let the packet in or out.
- **Bypass**: Do not apply or expect security.
- **Apply**: Apply security to outbound traffic and expect security for inbound traffic by determining a pointer to the SA in the SADB.

Selectors are extracted from the network and transport layer headers to map IP traffic to IPSec policy. IPSec policy selectors include source address, destination address, machine name, transport protocol, and source and destination TCP port numbers.

**IPSec Modes**

The AH and ESP header can be applied to an IP packet through modes defined by the IPSec architecture. There are several possible combinations of modes and protocols:

- **AH transport mode**
- **AH tunnel mode**
- **ESP transport mode**
- **ESP tunnel mode**

Transport mode applies AH and ESP to the transport layer segment of an IP packet. The IP data payload is the only protected portion of the packet. The IP header with the destination and source address information is not protected. When applying both AH and ESP, AH is applied last to calculate data integrity over more data.
Chapter 17: Security

On the other hand, the tunnel mode provides for authentication and encryption of the entire IP packet. Security gateways use tunnel mode to provide security services on behalf of other networked entities. The communication end point is protected inside the inner IP packet, while the crypto end point is contained in the outer IP packet. A security gateway de-encapsulates the inner IP packet upon completion of IPSec processing, then forwards the packet to its final destination. The end point IP address is protected in the tunnel.

Tunnel mode adds an extra IP Header (outer header), however transport mode does not add an extra IP header. IPSec defines tunnel mode for both AH and ESP. A tunnel using ESP is shown in Figure 95. Host 1 wanting to communicate with Host 2 can use tunnel mode to allow gateways SG1 and SG2 to provide the IPSec services to secure the communications over the public network. The inner IP header is the original packet from Host 1 destined for Host 2; the outer header encapsulates the original packet into another IP header.

Figure 94.  Transport Mode

Figure 95.  Tunnel Mode
IPSec allows for multiple layers of security and multiple tunnels in which the inner header is completely encompassed by the outer header. However, one tunnel cannot overlap another.

**Figure 96.** Multiple Tunnels

Figure 97 shows the packet constructed and seen by SG2. The packet is received and de-encrypted by the SG2 gateway. SG1, SG2, and finally Host 2 process the packet by de-encapsulating the headers from left to right until the original packet is left for processing by Host 2.

**Figure 97.** Received Packet

**Outbound Processing**

The IP layer consults the SPD to determine the security services to use. Selectors extracted from the headers are used to point to a policy action. If the SPD action is to apply security, a pointer to the SA in the SADB is returned, or IKE is invoked if the SA does not exist in the database. The AH and ESP headers are added as specified by the SA. The packet is forwarded as defined in the gateway or router.
Inbound Processing

Upon receipt of a packet, the security layer checks the policy database for these actions: discard, bypass, and apply. If the action required is apply and the SA does not exist, then the packet is dropped. However, if the SA exists in the database, then the packet is passed up to the next layer for processing. If the packet contains IPSec headers, then the IPSec stack processes the packets. IPSec extracts the SPI, source address, and destination address. The SADB is indexed based on the following parameters to select the specific SA to apply: SPI, DST Addr, or protocol (AH or ESP).

SSL/TLS

SSL creates a secure connection between a client and a server over which any amount of encrypted data can be sent. Web pages that require an SSL connection start their URLs with https instead of the normal http. However, these implementations require that both the sending and receiving stations run the required application software or Web browser—and only the data to and from the Web server is secured. SSL runs above TCP/IP and below high-level applications, such as Lightweight Directory Access Protocol (LDAP) and Internet Messaging Access Protocol (IMAP).

SSL is a protocol to secure web-based communications at the application layer using encryption and authentication. It allows users to define the level of security. Each application is secured one at a time. It has now been replaced by Transport Layer Security (TLS). TLS and SSL are not interoperable.

Each application server must support access via a web browser and each application must support SSL.

SSL Gateways serve as proxy to LAN applications such as e-mail and file servers.

MACSec

IEEE 802.1AE MACSec encrypts the entire frame and appends an Integrity Check Value (ICV). A LAN station receives the encrypted frame and calculates an ICV for the frame. If the ICVs match, then the frame is decrypted and handled normally, otherwise the frame may be discarded.

MACSec permits packet inspection at each participating LAN node, whereas IPSec tunnel mode encryption is only decrypted at the end station(s).

MACSec is an IEEE standard. It operates at L2 within switches and NICs. It is supported by the following standards:

- 802.1AE MACSec (Encryption)
- 802.1af MAC Key Security (Authentication – Key Distribution)
- 802.1AR Secure Device Identity (Authorization – Trusted Device)
As mentioned above, IEEE 802.1af MAC Key Security (Authentication – Key Distribution) establishes security associations for MACSec and manages short-lived session keys.

A master key may be obtained using 802.1X.

802.1AE will require dedicated encryption engines on each LAN node (switch, IDS, LB, etc.).
Protocols Comparison

Table 17 lists the comparison between IPSec and SSL encryption, deployment and application.

IPSec is best suited for users that require access to all LAN applications and resources with stronger security and authentication. It is likely to require hardware acceleration on the server end points.

SSL is best suited for users that need casual or mobile access to web enabled applications (extranets, e-mail, file share). A key difference is that IPSec is machine to machine and SSL/TLS is server to client (user or application). When you authenticate with IPSec, you are authenticating the machine and you are then trusting all applications and users on that machine. With SSL/TLS, you are authenticating the user.

Table 17. IPSec vs. SSL

<table>
<thead>
<tr>
<th>IPSec</th>
<th>SSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Provides stronger encryption</td>
<td>• Some implementations negotiate down to lowest common denominator encryption</td>
</tr>
<tr>
<td>• Ensures adherence to corporate security policies- devices must agree on the security association</td>
<td>• Must utilize two-factor authentication to guarantee identity of the user not the machine</td>
</tr>
<tr>
<td>• Pre-configured to know which server-side certificates to accept</td>
<td>• Browsers prompt the user to accept the certificate - an imposter can fool users into accepting a bogus certificate</td>
</tr>
<tr>
<td></td>
<td>• Information on the computer cache can be used by other users with access to the computer</td>
</tr>
<tr>
<td></td>
<td>• More complicated to deploy and manage</td>
</tr>
<tr>
<td></td>
<td>• Distribution of special purpose VPN client software</td>
</tr>
<tr>
<td></td>
<td>• Manual configuration of network and security parameters</td>
</tr>
<tr>
<td></td>
<td>• Configuration mistakes can compromise security</td>
</tr>
<tr>
<td></td>
<td>• Much easier to deploy</td>
</tr>
<tr>
<td></td>
<td>• Uses existing Browser software embedded in OS</td>
</tr>
<tr>
<td></td>
<td>• Requires applications that can accept SSL</td>
</tr>
<tr>
<td></td>
<td>• Existing network-accessible applications can be secured without modification</td>
</tr>
<tr>
<td></td>
<td>• Requires acceleration to maintain network performance at the end points</td>
</tr>
<tr>
<td></td>
<td>• Requires tight integration with the application</td>
</tr>
<tr>
<td></td>
<td>• Session oriented applications are difficult to integrate</td>
</tr>
<tr>
<td></td>
<td>• Acceleration required at the server</td>
</tr>
<tr>
<td></td>
<td>• TCO is lower since it does not require special client software, it is easier to deploy and manage</td>
</tr>
</tbody>
</table>
Stateful Offloads

As we have seen, the NIC is much more than an interconnect device. Most prevalent is the implementation of stateful offload protocols. The richness of capabilities enables fabric convergence on Ethernet via the multi-dimensional NIC, where the NIC supports multiple personalities concurrently in many cases.

Server operating systems support a variety of protocols in their stacks. They are classified as networking, storage, and MPI. The network APIs are based on Sockets over TCP/IP. The storage for block I/O API is SCSI for FCoE and iSCSI. MPI runs over RDMA Verbs with RoCE and iWARP.

Figure 100. Protocol Stacks
TOE

TCP Offload Engine (TOE) is a type of stateful offload that processes L3 (IP) and L4 (TCP) traffic on the NIC hardware, thereby freeing up compute cycles on the host CPU running the OS. This made sense when running at gigabit speeds on single CPU and limited memory systems. TOE offloaded the 3-way handshake connection establishment, data transfer and connection termination to the NIC. TOE was never widely adopted in the market place directly. The reason is that there are lots of CPU cores capable of handling the I/O traffic for most main stream server applications. Other concerns were related to security, feature support, and long term maintainability. However, TOE has found a home in iSCSI and iWARP.

Figure 101. TOE

A traditional NIC operates at L2 Ethernet level where TCP/IP runs in the host software stack as part of the OS kernel. A TOE NIC implements the TCP/IP stack down in the NIC.
Figure 102. NIC vs. TOE

TCP Chimney

TCP Chimney is a type of TOE partial offload that relies on the host TCP/IP networking stack to set up TCP connections, handle exceptions, provide IP address and TCP port assignments, and handle connection termination. It only offloads the data transfer to the NIC.

IPSec

IPSec is a suite of protocols that provides security for IP traffic at the network layer. It defines how to provide data integrity, authenticity, and confidentiality. It also allows enterprises to select specific security policy tailored to their needs. This is accomplished through tunneling, encryption, and authentication.

- Tunneling
  - Authentication Header (AH) or Encapsulating Security Payload (ESP)

- Encryption
  - 56-bit DES
  - 168-bit 3DES
  - 256-bit AES

- Authentication
  - Username and password (RADIUS)
  - Username and token
  - X.509 Digital certificates
End stations must agree to the same security policy and security association. An IPSEC tunnel secures all IP communications between stations.

- Traffic Type: TCP, UDP, SNMP
- Application: e-mail, client-server, database

Hosts require special-purpose software to create IPSec connections.

IPSec can be used to secure private communications over public networks. IPSec employs standards-based encryption and authentication and can be used to meet customer’s security and business needs.

IPSec is best suited for users that require access to all LAN applications and resources with stronger security and authentication. Hardware offload may be required on highly loaded end-points. IPSec can be offloaded to the hardware on the NIC in a bump-in-the-wire configuration. Typically, this includes encryption and authentication operations which are compute intensive. This can take the form of an ASIC or FPGA.

Storage

iSOE

iSCSI Offload Engine (iSOE) provides not only full TOE capability on the NIC but also provides iSCSI initiator functionality offloaded to the NIC running on top of TOE. The OS is not aware of the TOE; it only sees the SCSI device.

FCoE

FCoE (Fibre Channel over Ethernet) is a stateful offload that supports the transport of Fibre Channel protocol over Ethernet. The FCoE CNA exposes both a networking and storage device to the OS. It runs directly over Ethernet. There is no TCP/IP; therefore, it does not have benefits such as guaranteed delivery.

MPI

RoCEv1

RoCE (RDMA over Converged Ethernet) is a stateful offload, which supports RDMA verbs over a DCB capable Ethernet network. This version is not routable, since it runs directly on Ethernet and it has no IP.
RoCEv1 maintains the IB transport services and the existing OFED verbs API. At Layer 2, it replaces the IB LRH with the Ethernet MAC and the IB VCRC with the Ethernet FCS. RoCEv1 addressing relies on L3 IB addresses (GIDs).

**RoCEv2**

RDMA over Converged Ethernet (RoCE) is a stateful offload, which supports RDMA verbs over DCB capable Ethernet. RoCEv2 maintains the same IB transport services including Reliable/unreliable connected/datagram and support for Unicast and Multicast. It maintains the existing OFED verbs API. At Layer 2, it replaces the IB LRH with the Ethernet MAC and the IB VCRC with the Ethernet FCS. It replaces the IB GRH with UDP/IP to enable routability.

**Figure 103. RoCE and IB**

RoCEv1 maintains the IB transport services and the existing OFED verbs API. At Layer 2, it replaces the IB LRH with the Ethernet MAC and the IB VCRC with the Ethernet FCS. RoCEv1 addressing relies on L3 IB addresses (GIDs).

**RoCEv2**

RDMA over Converged Ethernet (RoCE) is a stateful offload, which supports RDMA verbs over DCB capable Ethernet. RoCEv2 maintains the same IB transport services including Reliable/unreliable connected/datagram and support for Unicast and Multicast. It maintains the existing OFED verbs API. At Layer 2, it replaces the IB LRH with the Ethernet MAC and the IB VCRC with the Ethernet FCS. It replaces the IB GRH with UDP/IP to enable routability.

**Figure 104. RoCE and RoCEv2 Frame**
iWARP

iWARP is a stateful offload that supports the transport of RDMA verbs over the TCP/IP stack to improve latency. It defines several wire protocols: RDMAP, DDP, and MPA. It is specified by the IETF RFC 5040, RFC 5041, and RFC 5044 respectively. These protocols enable the transfer of data between hosts on a network without intermediate CPU copies. The RNIC implements these protocols in its hardware.

![iWARP NIC Diagram](image)

**Figure 105.** iWARP NIC
The NIC will continue to evolve by adding new capabilities such as offload support for NVMe, NFV, and cloud infrastructure workloads. It will expand in new dimensions through SmartNICs that incorporate programmable elements to tackle new and complex tasks such as Machine Learning Algorithms.

The NIC controller ASIC will continue to add new applications such as NVMe over Fabrics using RDMA- RoCE or iWARP. This will allow remote NVMe media to expand beyond what is available in the local host. NFV will require hypervisor kernel bypass to enable the highest performing Virtual Network Function appliance. This will result in the use of DPDK/SR-IOV and full vSwitch offload to cut down the latency. SmartNICs will add Field Programmable Grid Array and/or ARM CPU multi-cores (MC) hardware to complement the existing, highly capable controller ASIC. The FPGA and MC will be preloaded with code that performs specific compute acceleration such as encryption and compression. In addition, customers will write and load their own code. This code could run High Frequency Trading algorithms, Big Data Analytics jobs, and Machine Learning algorithms among many others with the goal of improving compute performance.
### Abbreviation | Meaning
--- | ---
1GE | 1 Gigabit Ethernet
3DES | Triple DES
10GE | 10 Gigabit Ethernet
25GE | 25 Gigabit Ethernet
40GE | 40 Gigabit Ethernet
50GE | 50 Gigabit Ethernet
100GE | 100 Gigabit Ethernet
200GE | 200 Gigabit Ethernet
400GE | 400 Gigabit Ethernet
ACL | Access Control List
ACPI | Advanced Configuration and Power Interface (ACPI)
ACS | Access Control Services
AES | Advanced Encryption Standard
AH | Authentication Header
AN | Auto-Negotiation
API | Application Programming Interface
ARB | Arbitration
ARI | Alternative Routing-ID Interpretation
ARP | Address Resolution Protocol
ASIC | Application Specific Integrated Circuit
ASPM | Active State Power Management
ATS | Address Translation Services
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAR</td>
<td>Base Address Register</td>
</tr>
<tr>
<td>BDF</td>
<td>Bus/Device/Function</td>
</tr>
<tr>
<td>BIOS</td>
<td>Basic Input/Output System</td>
</tr>
<tr>
<td>BMC</td>
<td>Baseboard Management Controller</td>
</tr>
<tr>
<td>BPDUs</td>
<td>Bridge Protocol Data Units</td>
</tr>
<tr>
<td>BPE</td>
<td>Bridge Port Extension</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
</tr>
<tr>
<td>CAT</td>
<td>Category</td>
</tr>
<tr>
<td>CBC</td>
<td>Cipher Block Chaining</td>
</tr>
<tr>
<td>CDB</td>
<td>Command Descriptor Block</td>
</tr>
<tr>
<td>CEE</td>
<td>Converged Enhanced Ethernet</td>
</tr>
<tr>
<td>CHAP</td>
<td>Challenge-Handshake Authentication Protocol</td>
</tr>
<tr>
<td>CIM</td>
<td>Common Information Model</td>
</tr>
<tr>
<td>CLP</td>
<td>Command Line Protocol</td>
</tr>
<tr>
<td>CN</td>
<td>Congestion Notification</td>
</tr>
<tr>
<td>CNA</td>
<td>Converged Network Adapter</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CSMA/CD</td>
<td>Carrier Sense Multiple Access/Collision Detection</td>
</tr>
<tr>
<td>CSO</td>
<td>Checksum Offload</td>
</tr>
<tr>
<td>DA</td>
<td>Discovery Advertisement</td>
</tr>
<tr>
<td>DAC</td>
<td>Direct Attach Copper</td>
</tr>
<tr>
<td>DCB</td>
<td>Data Center Bridging</td>
</tr>
<tr>
<td>DCBX</td>
<td>Data Center Bridging Exchange Protocol</td>
</tr>
<tr>
<td>DDP</td>
<td>Direct Data Placement</td>
</tr>
<tr>
<td>DES</td>
<td>Data Encryption Standard</td>
</tr>
<tr>
<td>DLLP</td>
<td>Data Link Layer Packet</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>DMTF</td>
<td>Distributed Management Task Force</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name Services</td>
</tr>
<tr>
<td>DPC</td>
<td>Deferred Procedure Call</td>
</tr>
<tr>
<td>DPDK</td>
<td>Data Plane Development Kit</td>
</tr>
<tr>
<td>DRAM</td>
<td>Dynamic Random-Access Memory</td>
</tr>
<tr>
<td>DRS</td>
<td>Distributed Resource Scheduler</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>DS</td>
<td>Discovery Solicitation</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>DSQ</td>
<td>Double Square QAM</td>
</tr>
<tr>
<td>DTE</td>
<td>Data Terminal Equipment</td>
</tr>
<tr>
<td>ECN</td>
<td>Engineering Change Notice</td>
</tr>
<tr>
<td>EDR</td>
<td>Extended Data Rate</td>
</tr>
<tr>
<td>EEE</td>
<td>Energy Efficient Ethernet</td>
</tr>
<tr>
<td>EEPROM</td>
<td>Electrically erasable programmable read-only memory</td>
</tr>
<tr>
<td>EPT</td>
<td>Extended Page Table</td>
</tr>
<tr>
<td>e-Switch</td>
<td>Embedded Switch</td>
</tr>
<tr>
<td>EMI</td>
<td>Electromagnetic Interference</td>
</tr>
<tr>
<td>ESP</td>
<td>Encapsulating Security Payload</td>
</tr>
<tr>
<td>ETS</td>
<td>Enhanced Transmission Selection</td>
</tr>
<tr>
<td>EVB</td>
<td>Edge Virtual Bridge</td>
</tr>
<tr>
<td>FC</td>
<td>Fibre Channel</td>
</tr>
<tr>
<td>FCP</td>
<td>Fibre Channel Protocol</td>
</tr>
<tr>
<td>FCoE</td>
<td>Fibre Channel over Ethernet</td>
</tr>
<tr>
<td>FCF</td>
<td>Fibre Channel Forwarder</td>
</tr>
<tr>
<td>FCS</td>
<td>Frame Check Sequence</td>
</tr>
<tr>
<td>FDF</td>
<td>FCoE Data Forwarder</td>
</tr>
<tr>
<td>FDISC</td>
<td>Fabric Discovery</td>
</tr>
<tr>
<td>FDR</td>
<td>Fourteen Data Rate</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>FIP</td>
<td>FCoE Initialization Protocol</td>
</tr>
<tr>
<td>FLP</td>
<td>Fast Link Pulse</td>
</tr>
<tr>
<td>FLOGI</td>
<td>Fabric Login</td>
</tr>
<tr>
<td>FLOGO</td>
<td>Fabric Logout</td>
</tr>
<tr>
<td>FLR</td>
<td>Function Level Reset</td>
</tr>
<tr>
<td>FPMA</td>
<td>Fabric Provided MAC Address</td>
</tr>
<tr>
<td>FT</td>
<td>Fault Tolerance</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>FQDD</td>
<td>Fully Qualified Device Descriptor</td>
</tr>
<tr>
<td>FQDN</td>
<td>Fully Qualified Domain Name</td>
</tr>
<tr>
<td>G-ARP</td>
<td>Gratuitous ARP</td>
</tr>
<tr>
<td>Gb/s or Gbps</td>
<td>Gigabit per second</td>
</tr>
<tr>
<td>GE</td>
<td>Gigabit Ethernet</td>
</tr>
<tr>
<td>GID</td>
<td>Global Identifier</td>
</tr>
<tr>
<td>GOS</td>
<td>Guest Operating System</td>
</tr>
<tr>
<td>GRH</td>
<td>Global Route Header</td>
</tr>
<tr>
<td>HBA</td>
<td>Host Bus Adapter</td>
</tr>
<tr>
<td>HCA</td>
<td>Host Controller Adapter</td>
</tr>
<tr>
<td>HDD</td>
<td>Hard Disk Drive</td>
</tr>
<tr>
<td>HFI</td>
<td>Host Fabric Interface</td>
</tr>
<tr>
<td>HII</td>
<td>Human Infrastructure Interface</td>
</tr>
<tr>
<td>HPC</td>
<td>High Performance Computing</td>
</tr>
<tr>
<td>HFT</td>
<td>High Frequency Trading</td>
</tr>
<tr>
<td>HMAC</td>
<td>Hashed Message Authentication Code</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>I2C</td>
<td>Inter-Integrated Circuit</td>
</tr>
<tr>
<td>IB</td>
<td>Infiniband</td>
</tr>
<tr>
<td>iBFT</td>
<td>iSCSI Boot Firmware Table</td>
</tr>
<tr>
<td>ICV</td>
<td>Integrity Check Value</td>
</tr>
<tr>
<td>IHV</td>
<td>Independent Hardware Vendor</td>
</tr>
<tr>
<td>IKE</td>
<td>Internet Key Exchange</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>IOMMU</td>
<td>Input–output memory management unit</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPSec</td>
<td>IP Security</td>
</tr>
<tr>
<td>IPX</td>
<td>Internetwork Packet Exchange</td>
</tr>
<tr>
<td>iSCSI</td>
<td>Internet Small Computer System Interface</td>
</tr>
<tr>
<td>ISAKMP</td>
<td>Internet Security Association and Key Management Protocol</td>
</tr>
<tr>
<td>iSNS</td>
<td>Internet Storage Name Service</td>
</tr>
<tr>
<td>iSOE</td>
<td>iSCSI Offload Engine</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>IV</td>
<td>Initialization Vector</td>
</tr>
<tr>
<td>KR</td>
<td>Copper Backplane based on the IEEE 802.3ap specifications</td>
</tr>
<tr>
<td>LACP</td>
<td>Link Aggregation Control Protocol</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LB</td>
<td>Load Balancing</td>
</tr>
<tr>
<td>LBFO</td>
<td>Load Balancing and Failover</td>
</tr>
<tr>
<td>LC</td>
<td>Lucent Connector (Optical)</td>
</tr>
<tr>
<td>LCC</td>
<td>Lifecycle controller</td>
</tr>
<tr>
<td>LCW</td>
<td>Link Code Word</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>LDPC</td>
<td>Low Density Parity Check</td>
</tr>
<tr>
<td>LIT</td>
<td>Link Integrity Test</td>
</tr>
<tr>
<td>LLDP</td>
<td>Link Layer Discovery Protocol</td>
</tr>
<tr>
<td>LOM</td>
<td>LAN on Motherboard</td>
</tr>
<tr>
<td>LPI</td>
<td>Low Power Idle</td>
</tr>
<tr>
<td>LRO</td>
<td>Large Receive Offload</td>
</tr>
<tr>
<td>LSO</td>
<td>Large Send Offload</td>
</tr>
<tr>
<td>LU</td>
<td>Logical unit</td>
</tr>
<tr>
<td>LUN</td>
<td>Logical unit number</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Controller</td>
</tr>
<tr>
<td>MACSec</td>
<td>MAC Security</td>
</tr>
<tr>
<td>MDI</td>
<td>Medium Dependent Interface</td>
</tr>
<tr>
<td>MD5</td>
<td>Message Digest 5</td>
</tr>
<tr>
<td>MDIO</td>
<td>Management Data Input/Output</td>
</tr>
<tr>
<td>MIB</td>
<td>Management Information Base</td>
</tr>
<tr>
<td>MMF</td>
<td>Multi-mode Fiber</td>
</tr>
<tr>
<td>MMIO</td>
<td>Memory Mapped IO</td>
</tr>
<tr>
<td>MLAG</td>
<td>Multi-chassis Link Aggregation</td>
</tr>
<tr>
<td>MLD</td>
<td>Multi-Lane Distribution</td>
</tr>
<tr>
<td>MSI</td>
<td>Message Signaled Interrupt</td>
</tr>
<tr>
<td>MSS</td>
<td>Maximum Segment Size</td>
</tr>
<tr>
<td>MTU</td>
<td>Maximum Transmission Unit</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>NC-SI</td>
<td>Network Controller Sideband Interface</td>
</tr>
<tr>
<td>NDC</td>
<td>Network Daughter Card</td>
</tr>
<tr>
<td>NDIS</td>
<td>Network Driver Interface Specification</td>
</tr>
<tr>
<td>NDKPI</td>
<td>Network Direct Kernel Programming Interface</td>
</tr>
<tr>
<td>NetDev</td>
<td>Network Device (Driver Interface configuration)</td>
</tr>
<tr>
<td>NFV</td>
<td>Network Function Virtualization</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>NLP</td>
<td>Normal Link Pulse</td>
</tr>
<tr>
<td>NPT</td>
<td>Nested Page Table</td>
</tr>
<tr>
<td>NPar</td>
<td>NIC Partitioning</td>
</tr>
<tr>
<td>NUMA</td>
<td>Non-uniform memory access</td>
</tr>
<tr>
<td>NVGRE</td>
<td>Network Virtualization using Generic Routing Encapsulation</td>
</tr>
<tr>
<td>NVMe</td>
<td>Non-Volatile Memory over PCIe</td>
</tr>
<tr>
<td>NVRAM</td>
<td>Non-Volatile Random Access Memory</td>
</tr>
<tr>
<td>OCP</td>
<td>Open Compute Platform</td>
</tr>
<tr>
<td>OFED</td>
<td>Open Fabrics Enterprise Distribution</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnect</td>
</tr>
<tr>
<td>OUI</td>
<td>Organizational Unique Identifier</td>
</tr>
<tr>
<td>OVS</td>
<td>OpenvSwitch</td>
</tr>
<tr>
<td>PCI</td>
<td>Peripheral Component Interconnect</td>
</tr>
<tr>
<td>PCIe</td>
<td>PCI Express</td>
</tr>
<tr>
<td>PCS</td>
<td>Physical Coding Sublayer</td>
</tr>
<tr>
<td>PDPI</td>
<td>Packet Direct Provider Interface</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Units</td>
</tr>
<tr>
<td>PFC</td>
<td>Priority-based Flow Control</td>
</tr>
<tr>
<td>PHY</td>
<td>Physical Layer</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase Locked Loop</td>
</tr>
<tr>
<td>PLOGI</td>
<td>Port Login</td>
</tr>
<tr>
<td>PMA</td>
<td>Physical Medium Attachment</td>
</tr>
<tr>
<td>PMD</td>
<td>Physical Medium Dependent</td>
</tr>
<tr>
<td>PTP</td>
<td>Precision Timing Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>PXE</td>
<td>Preboot Execution Environment</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>QSFP+</td>
<td>Quad SFP+ (4x10 Gbps)</td>
</tr>
<tr>
<td>QSFP28</td>
<td>Quad SFP (4x28 Gbps)</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Array of Inexpensive Disks</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RDMA</td>
<td>Remote DMA</td>
</tr>
<tr>
<td>RJ-45</td>
<td>Registered Jack</td>
</tr>
<tr>
<td>RMII</td>
<td>Reduced Media Independent Interface</td>
</tr>
<tr>
<td>RNIC</td>
<td>RDMA NIC</td>
</tr>
<tr>
<td>RoCE</td>
<td>RDMA over Converged Ethernet</td>
</tr>
<tr>
<td>RRoCE</td>
<td>Routable RoCE</td>
</tr>
<tr>
<td>ROM</td>
<td>Read Only Memory</td>
</tr>
<tr>
<td>RSS</td>
<td>Receive Side Scaling</td>
</tr>
<tr>
<td>RSC</td>
<td>Receive Side Coalescing</td>
</tr>
<tr>
<td>SA</td>
<td>Security Association</td>
</tr>
<tr>
<td>SADB</td>
<td>Security Association Database</td>
</tr>
<tr>
<td>SAN</td>
<td>Storage Area Network</td>
</tr>
<tr>
<td>SCSI</td>
<td>Small Computer System Interface</td>
</tr>
<tr>
<td>SET</td>
<td>Switch Extended Teaming</td>
</tr>
<tr>
<td>SFI</td>
<td>SerDes Framer Interface</td>
</tr>
<tr>
<td>SFP</td>
<td>Small Form-factor Pluggable (1 Gbps Connector)</td>
</tr>
<tr>
<td>SFP+</td>
<td>Small Form-factor Pluggable Plus (10 Gbps Connector)</td>
</tr>
<tr>
<td>SFP28</td>
<td>Small Form-factor Pluggable 28 (25 Gbps Connector)</td>
</tr>
<tr>
<td>SFP56</td>
<td>Small Form-factor Pluggable 50 (50 Gbps Connector)</td>
</tr>
<tr>
<td>SHA</td>
<td>Secure Hash Algorithm</td>
</tr>
<tr>
<td>S-HTTP</td>
<td>Secure HTTP</td>
</tr>
<tr>
<td>SIG</td>
<td>Special Interest Group</td>
</tr>
<tr>
<td>SLES</td>
<td>SUSE Linux Enterprise Server</td>
</tr>
<tr>
<td>SMB</td>
<td>Server Message Block</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>SMBus</td>
<td>System Management Bus</td>
</tr>
<tr>
<td>SMF</td>
<td>Single-mode Fiber</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>SPB</td>
<td>Shortest Path Bridging</td>
</tr>
<tr>
<td>SPD</td>
<td>Security Policy Database</td>
</tr>
<tr>
<td>SPI</td>
<td>Security Parameters Index</td>
</tr>
<tr>
<td>SPMA</td>
<td>Server Provided MAC Address</td>
</tr>
<tr>
<td>STag</td>
<td>Steering Tag</td>
</tr>
<tr>
<td>STP</td>
<td>Spanning Tree Protocol</td>
</tr>
<tr>
<td>STP</td>
<td>Shielded Twisted Pair</td>
</tr>
<tr>
<td>SR</td>
<td>Short Range (Optical Transceiver)</td>
</tr>
<tr>
<td>SR-IOV</td>
<td>Single Root I/O Virtualization</td>
</tr>
<tr>
<td>SSD</td>
<td>Solid State Drive</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>TAF</td>
<td>Technology Ability Field</td>
</tr>
<tr>
<td>TC</td>
<td>Traffic Class</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TFTP</td>
<td>Trivial File Transfer Protocol</td>
</tr>
<tr>
<td>TLB</td>
<td>Transaction Lookaside Buffer</td>
</tr>
<tr>
<td>TLP</td>
<td>Transaction Layer Processing</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>TLV</td>
<td>Type, Length, Value</td>
</tr>
<tr>
<td>TNI</td>
<td>Tenant network ID</td>
</tr>
<tr>
<td>TOE</td>
<td>TCP Offload Engine</td>
</tr>
<tr>
<td>TRILL</td>
<td>Transparent Interconnection of Lots of Links</td>
</tr>
<tr>
<td>TSO</td>
<td>TCP Segmentation Offload</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>UEFI</td>
<td>Unified Extensible Firmware Interface</td>
</tr>
<tr>
<td>ULP</td>
<td>Upper Level Protocol</td>
</tr>
<tr>
<td>UNDI</td>
<td>Universal Network Driver Interface</td>
</tr>
<tr>
<td>UTP</td>
<td>Unshielded Twisted Pair</td>
</tr>
<tr>
<td>VC</td>
<td>Virtual Channel</td>
</tr>
<tr>
<td>VEB</td>
<td>Virtual Ethernet Bridge</td>
</tr>
<tr>
<td>VF</td>
<td>Virtual Function</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>VLAN</td>
<td>Virtual LAN</td>
</tr>
<tr>
<td>VLT</td>
<td>Virtual Link Trunking</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>VMM</td>
<td>VM Manager</td>
</tr>
<tr>
<td>VMQ</td>
<td>VM Queues</td>
</tr>
<tr>
<td>VNF</td>
<td>Virtual Network Function</td>
</tr>
<tr>
<td>VNI</td>
<td>Virtual Network ID</td>
</tr>
<tr>
<td>vNIC</td>
<td>Virtual NIC</td>
</tr>
<tr>
<td>vPC</td>
<td>Virtual Port Channel</td>
</tr>
<tr>
<td>vSAN</td>
<td>Virtual SAN</td>
</tr>
<tr>
<td>vSwitch</td>
<td>Virtual Switch</td>
</tr>
<tr>
<td>VT-d</td>
<td>Virtual Technology for Directed IO</td>
</tr>
<tr>
<td>VxLAN</td>
<td>Virtual Extensible LAN</td>
</tr>
<tr>
<td>WINS</td>
<td>Windows Internet Name Services</td>
</tr>
<tr>
<td>WWNN</td>
<td>World Wide Node Name</td>
</tr>
<tr>
<td>WWPN</td>
<td>World Wide Port Name</td>
</tr>
<tr>
<td>XAUI</td>
<td>10GE Attachment Unit Interface</td>
</tr>
<tr>
<td>XGMII</td>
<td>10GE Media Independent Interface</td>
</tr>
</tbody>
</table>
Appendix B

References

- DMTF, DSP0222 Network Controller Sideband Interface (NC-SI), 1.0.1
- IEEE, IEEE Std 802.1D Media Access Control (MAC) Bridges, 2004
- IEEE, IEEE Std 802.1Q Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks, 2011
- IEEE, IEEE Std 802.1Qaz Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks, Amendment 18: Enhanced Transmission Selection for Bandwidth Sharing Between Traffic Classes, 2011
- IEEE, IEEE Std 802.1Qbb Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks, Amendment 17: Priority-based Flow Control, 2011
- PCI-SIG, PCI Express Base Specification 1.1
- PCI-SIG, PCI Express Base Specification 2.0
- PCI-SIG, PCI Express Base Specification 3.0
- PCI-SIG, Single Root I/O Virtualization and Sharing, 1.1


RFC2890- Key and Sequence Number Extensions to GRE

RFC 7637 on NVGRE: Network Virtualization Using Generic Routing Encapsulation

Virtual eXtensible Local Area Network (VxLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks


Rich Hernandez. *Improving e-business server availability.* Computer Technology Review; Feb 2001; 21, 2; ABI/INFORM Complete page 28

Rich Hernandez. *Dell PowerEdge SRIOV Host LAN Device Requirements.* January 2010

---

172 ♦ Appendix B: References
- Rich Hernandez. *Dell FCoE Host Adapter Requirements*. August 2010
- Rich Hernandez. *Dell Datacenter Bridging Host LAN Device Requirements*. July 2010
- Sandhya Senapathi and Rich Hernandez. *Introduction to TCP Offload Engines*. Power Solutions, March 2004
Appendix C

Works Cited


- Submitted to American Intercontinental University Online on 2014-01-23


http://www.emulex.com/artifacts/e51497f0-a678-466a-8ea4-6724bb3dbff/elx_wp_all_10GbeGlossary.pdf


Figures

- **Figure 6:** The Tech FAQ – [https://www.google.com/search?q=osimodel+images&espv=2&biw=1280&bih=617&tbm=isch&imgil=6yxO6UlrfkeT6M%253A%253B%2535ajDZwA2k771M%253Bhttp%25252F%25252Fwww.tech-faq.com%25252Fosi-model.html&source=iu&pf=m&fir=6yxO6UlrfkeT6M%253A%253B%2535ajDZwA2k771M%253B%253Dv=0ahUKEwi4obWoo6zRAhVM9GMKHQxvAwEQyjcINQ&ei=huzuWlJGLMzojwOM3oI#imgrc=6yxO6UlrfkeT6M%3A](https://www.google.com/search?q=osimodel+images&espv=2&biw=1280&bih=617&tbm=isch&imgil=6yxO6UlrfkeT6M%253A%253B%2535ajDZwA2k771M%253Bhttp%25252F%25252Fwww.tech-faq.com%25252Fosi-model.html&source=iu&pf=m&fir=6yxO6UlrfkeT6M%253A%253B%2535ajDZwA2k771M%253B%253Dv=0ahUKEwi4obWoo6zRAhVM9GMKHQxvAwEQyjcINQ&ei=huzuWlJGLMzojwOM3oI#imgrc=6yxO6UlrfkeT6M%3A)

- **Figure 7:** HowStuffWorks PCIe SIG – [https://www.google.com/search?q=osimodel+images&espv=2&biw=1280&bih=617&tbm=isch&imgil=6yxO6UlrfkeT6M%253A%253B%2535ajDZwA2k771M%253Bhttp%25252F%25252Fwww.tech-faq.com%25252Fosi-model.html&source=iu&pf=m&fir=6yxO6UlrfkeT6M%253A%253B%2535ajDZwA2k771M%253B%253Dv=0ahUKEwi4obWoo6zRAhVM9GMKHQxvAwEQyjcINQ&ei=huzuWlJGLMzojwOM3oI#imgrc=EJWXkFcIywUaCM%3A](https://www.google.com/search?q=osimodel+images&espv=2&biw=1280&bih=617&tbm=isch&imgil=6yxO6UlrfkeT6M%253A%253B%2535ajDZwA2k771M%253Bhttp%25252F%25252Fwww.tech-faq.com%25252Fosi-model.html&source=iu&pf=m&fir=6yxO6UlrfkeT6M%253A%253B%2535ajDZwA2k771M%253B%253Dv=0ahUKEwi4obWoo6zRAhVM9GMKHQxvAwEQyjcINQ&ei=huzuWlJGLMzojwOM3oI#imgrc=EJWXkFcIywUaCM%3A)

- **Figure 8:** QLogic* – Slide deck

- **Figure 19:** DMTF – DSP0222

- **Figure 21:** IEEE 802.3 STD

- **Figure 22:** Robert Winter – Dell

- **Figure 23:** IEEE – IEEE 802.3 STD

- **Figure 24:** IEEE – IEEE 802.3 STD

- **Figure 25:** Robert Winter – Dell

- **Figure 26:** IEEE – IEEE 802.3 STD

- **Figure 27:** IEEE – IEEE 802.3 STD

- **Figure 28:** IEEE – IEEE 802.3 STD

- **Figure 29:** IEEE – IEEE 802.3 STD

- **Figure 30:** Dell

- **Figure 31:** Microsoft – iSCSI Firmware Assisted SAN Boot

- **Figure 51:** IETF – Virtual eXtensible Local Area Network (VxLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks RFC 7348

- **Figure 52:** IETF – NVGRE: Network Virtualization Using Generic Routing Encapsulation (RFC 7637, September 2015)
- Figure 53: VMWare – VMware Virtual Networking Concepts
- Figure 54: VMWare – VMware Virtual Networking Concepts
- Figure 55: PCI SIG – Single Root I/O Virtualization and Sharing Specification
- Figure 60: IETF – Internet Small Computer Systems Interface (iSCSI) From RFC 3720
- Figure 64: INCITS T11 – FC-BB-5, the T11 standard for FCoE was ratified in June 2009
- Figure 66: INCITS T11 – FC-BB-5, the T11 standard for FCoE was ratified in June 2009
- Figure 67: QLogic*, Emulex – Slide presentations
- Figure 72: IEEE – IEEE 802.3 STD
- Figure 73: IEEE – IEEE802.1 STD
- Figure 74: IEEE – IEEE802.1 STD
- Figure 75: IEEE – IEEE802.1 STD
- Figure 76: IEEE – IEEE802.1 STD
- Figure 78: Microsoft – http://www.bing.com/images/search?q=smbdirect&view=detailv2&id=8BA9123FF023C17C35F5151A24CEED00D5DB023E&selectedIndex=22&ccid=NfE7tlqq&simid=608046613119241613&thid=OIP.M35f13bb65aaa872fda8fff272d258163o0&ajaxhist=0

Tables

- Table 7: IEEE – IEEE 802.3 STD
- Table 8: IEEE – IEEE 802.3 STD
- Table 11: IEEE – IEEE802.1 STD
- Table 15: IEEE – IEEE 802.3 STD

* QLogic is now Cavium FastLinQ
Index

A
ACPI
  boot configuration parameters 69
ACS 98
AES
  Encryption 155
AH. See SA
  about 142
  data integrity calculation 147
  header extension 140
API
  function calls 3
  OS API specifications 2
  RDMA API 133
  Sockets API 27
ARI
  about 98
  PCI functions 102
  SR-IOV 101
ARP. See G-ARPs
  about 80
  ARP cache entries 77
  ARP request and ARP reply frame exchange 27
  destination MAC address 27
ASIC
  about 5
ASPM
  about 34
Auto-Neg
  link speed setting 1
  master-slave mode 46

B
BAR
  about 33
BDF 32
  traditional PCI BDF routing ID 101
BIOS
  legacy BIOS boot mode 70
  SR-IOV devices 99
BMC
  defined 9
  NC-SI for pass-thru communication 16
  SMBus interface 15

C
CEE 119
CNA
  about 3
  configuration 70
  FCoE 156
CPU
  CPU memory copy and CPU context switch 22
  host CPU 56
  PCIe 31
  reduces CPU utilization and latency 128
  reduce substantial overhead of 85
  RSS 87
<table>
<thead>
<tr>
<th>Page</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td>CRC checksum</td>
</tr>
<tr>
<td>52</td>
<td>CRC checksum errors</td>
</tr>
<tr>
<td>30</td>
<td>CRC checksum field</td>
</tr>
<tr>
<td>86</td>
<td>CSO host CPU utilization</td>
</tr>
<tr>
<td>85</td>
<td>TCP/IP CSO technique</td>
</tr>
<tr>
<td>62</td>
<td>DAC SFP support</td>
</tr>
<tr>
<td>65</td>
<td>DAC ToR switch</td>
</tr>
<tr>
<td>41</td>
<td>DAC twinax cables</td>
</tr>
<tr>
<td>15</td>
<td>Data Link Layer about</td>
</tr>
<tr>
<td>53</td>
<td>DCB about</td>
</tr>
<tr>
<td>132</td>
<td>DCB PFC, ETS, and DCBX</td>
</tr>
<tr>
<td>53</td>
<td>DCB support for multiple traffic classes</td>
</tr>
<tr>
<td>119</td>
<td>DCB support lossless flows</td>
</tr>
<tr>
<td>123</td>
<td>DCBX ETS</td>
</tr>
<tr>
<td>142</td>
<td>DCBX DES about</td>
</tr>
<tr>
<td>131</td>
<td>Direct Data Placement about</td>
</tr>
<tr>
<td>135</td>
<td>Direct Data Placement buffer model</td>
</tr>
<tr>
<td>135</td>
<td>Direct Data Placement ULP messages</td>
</tr>
<tr>
<td>25</td>
<td>DMA about</td>
</tr>
<tr>
<td>57</td>
<td>DMA controller</td>
</tr>
<tr>
<td>95</td>
<td>DMA independent DMA streams, virtual functions</td>
</tr>
<tr>
<td>22</td>
<td>DMA packet flow</td>
</tr>
<tr>
<td>85</td>
<td>DMA transfer of data packets</td>
</tr>
<tr>
<td>27</td>
<td>DNS about</td>
</tr>
<tr>
<td>22</td>
<td>DPC about</td>
</tr>
<tr>
<td>86</td>
<td>DPC DPDK defined</td>
</tr>
<tr>
<td>129</td>
<td>DPDK DSP defined</td>
</tr>
<tr>
<td>1</td>
<td>DSP about</td>
</tr>
<tr>
<td>62</td>
<td>DSP techniques</td>
</tr>
<tr>
<td>59</td>
<td>E EEE about</td>
</tr>
<tr>
<td>3</td>
<td>EEPROM pre-OS operations</td>
</tr>
<tr>
<td>32</td>
<td>EMI about</td>
</tr>
<tr>
<td>140</td>
<td>ESP about</td>
</tr>
<tr>
<td>141</td>
<td>ETS about</td>
</tr>
<tr>
<td>119</td>
<td>ETS, 123</td>
</tr>
<tr>
<td>156</td>
<td>EEE about</td>
</tr>
<tr>
<td>32</td>
<td>EEPROM about</td>
</tr>
<tr>
<td>40</td>
<td>EEE about</td>
</tr>
<tr>
<td>61</td>
<td>EEE about</td>
</tr>
<tr>
<td>119</td>
<td>ESXi about</td>
</tr>
<tr>
<td>123</td>
<td>ESXi about</td>
</tr>
<tr>
<td>113</td>
<td>ELAN about</td>
</tr>
<tr>
<td>114</td>
<td>F FCoE guaranteed delivery</td>
</tr>
<tr>
<td>119</td>
<td>FCoE quality of service</td>
</tr>
<tr>
<td>119</td>
<td>FCoE transport of</td>
</tr>
<tr>
<td>110</td>
<td>FCoE Fibre Channel protocol over Ethernet</td>
</tr>
<tr>
<td>114</td>
<td>FCoE command execution</td>
</tr>
<tr>
<td>115</td>
<td>FCoE discovery using FIP</td>
</tr>
<tr>
<td>110</td>
<td>FCoE FIP about</td>
</tr>
<tr>
<td>110</td>
<td>FCoE FCF about</td>
</tr>
<tr>
<td>58</td>
<td>FEC 802.3ap (Backplane Ethernet) about</td>
</tr>
<tr>
<td>40</td>
<td>FEC auto-negotiation</td>
</tr>
<tr>
<td>61</td>
<td>FEC enable acceptable bit error rates</td>
</tr>
<tr>
<td>22</td>
<td>FIFO frame buffering</td>
</tr>
<tr>
<td>21</td>
<td>FIFO packet buffer</td>
</tr>
<tr>
<td>113</td>
<td>FIP about</td>
</tr>
<tr>
<td>110</td>
<td>FIP FCF about</td>
</tr>
<tr>
<td>110</td>
<td>FIP FCoE Initialization Protocol (FIP) and Encapsulation operations</td>
</tr>
<tr>
<td>110</td>
<td>FIP MAC addresses used in FIP</td>
</tr>
</tbody>
</table>
Firecode FEC
   lower latency FEC 41

FLP
   auto-negotiation 49
   base link code word (LCW) 50

FLR
   device reset at a per VF level 99

IPSec
   about 139
      characteristics 140
      SA 146
iSCSI
   iSCSI PDUs 107
   iSOE 105
   remote disk 69

H

HFT
   latency 30

HMAC
   Authentication code 142
      mandatory authenticators 144

Host Bus Adapter
   about 19

I

iBFT
   iSCSI Configuration Block
      parameters 70
   iSCSI Option ROM 69

IKE
   authentication 144
   Internet Security Association and
      Key Management Protocol
         (ISAKMP) 140
   negotiate an IPSec SA 144
   request-response protocol 145

I/O
   direct I/O assignment 93
   I/O architectural options 93
   iSCSI Initiators, iSCSI Targets 105

IP
   ARP 27
   data security 139
   FCoEv1 110
   IP address on another subnet 28
   layer 3 protocols 27
   socket 27
   tagged Virtual LAN (VLAN) 18
   TOE 154
M

MAC
ARP  27
ASIC  5
Ethernet frame  30
L2 MAC address  24
Layer 2 address  26
NPar  101
unique MAC address  77
MSI
about  34
MTU
defined  52

N

NIC
data link layer  1
dMA  25
hardware offloads  106
host CPU  5
interface to the host  2
Multiple NICs  74
network stack  67
NPar  99
sends and receives frames  21
system resources  56
vNICS  94
Wake on LAN  34

NVGRE
large cloud computing
deployments  89
overlay tunnels  89

O

OptionROM
initializing associated hardware  67
ROM chip  67
OS
API specifications  2
host memory set up by  23
host software stack  154
NIC teaming  81

OSI
NIC  1

P

PAM4
encoding techniques  61
PCIe
connects CPUs to peripheral
devices  31
ID  15
SR-IOV  19
standards  15
traffic classes  122
PHY
data communications  5
KR or KR4 PHY  16
media dependent interface  57
mezzanine card  66
PXE
operating system download  67

R

RDMA
API  133
DCB  53
eliminate memory copies  131
RoCE  156
zero copy and OS stack bypass  131
RNIC
RDMA Protocol Verbs
Specification  133
RSC
network packets processing  86
RSS
network processing load  87

S

SA
deletion  146
IKE  144
IPSec  146
SPI  141
SADB
  SA parameters  146
SHA
  data integrity  142
SMB
  RDMA transport  128
SR-IOV
  natively shareable device  95
  virtual function  99
SSL
  secure connection  150

T
TCP
  connections  107
TOE
  stateful offload  154

U
UDP
  UDP port number  133
UEFI
  driver  71
UNDI
  iSCSI boot ROM  69

V
VEB
  Virtual Ethernet Bridge  93
VLAN
  tagging  83
  tag insertion  82
  tags  83
VxLAN
  header, Virtual Network ID  18

W
WoL
  wake-up event  45
At the Network’s Edge will help you understand the evolution of the network interface card and obtain a broader view of the server networking subsystem. This book will instill in you a deeper appreciation for the rich and diverse capabilities offered by the data communications protocol stack manifested by the NIC at the edge of the network.

You will get an in-depth insight into the components of the host networking ecosystem that includes the operating system networking stack, the PCI Express host interface, and the local area network. Through this book, you will be able to navigate the various dimensions where the NIC resides. In addition, it will serve as a guide that highlights the key NIC attributes along each of these dimensions, which include the NIC environmental dimensions; the key software, firmware, and hardware components; and the operational planes of Data, Management, and Control.