IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

ALACRITECH INC., a California corporation, Plaintiff, v. WISTRON CORPORATION, a Taiwanese corporation, WIWYNN CORPORATION, a Taiwanese corporation, SMS INFOCOMM CORPORATION, a Texas corporation, Defendants.	Case No. 2:16-cv-692 **Jury Trial Demanded**

COMPLAINT FOR PATENT INFRINGEMENT

In this action for patent infringement under 35 U.S.C. § 271, Plaintiff Alacritech Inc. ("Alacritech"), by and through its undersigned counsel, complains and alleges as follows against Defendants Wistron Corporation ("Wistron"), Wiwynn Corporation ("Wiwynn"), and SMS InfoComm Corporation ("SMS InfoComm") (collectively referred to herein as "Defendants"), based on Alacritech's own personal knowledge with respect to its own actions and upon information and belief with respect to others' actions:

THE PARTIES

1. Alacritech is a California corporation with its principal place of business at P.O. Box 20307, San Jose, California 95160.

2. Wistron Corporation is a Taiwanese corporation, with its principal place of business at 158, Singshan Road, Neihu, Taipei 11469, Taiwan, R.O.C.

3. Wiwynn Corporation is a Taiwanese corporation and a subsidiary of Wistron Corporation, with its principal place of business at 8F, 90, Section 1, Xintai 5th Road, Xizhi District, New Taipei City 22102, Taiwan, R.O.C.

4. SMS InfoComm Corporation is a Texas corporation and a subsidiary of Wistron Corporation, with its principal place of business at 4051 North Highway 121, Suite 100, Grapevine, TX 76051.

NATURE OF THE ACTION

5. This is a civil action for patent infringement arising under the patent laws of the United States, 35 U.S.C. § 1, *et seq*.

6. Defendants have infringed and continue to infringe, have contributed to and continue to contribute to the infringement of, and have actively induced and continue to actively induce others to infringe Alacritech's U.S. Patent Numbers 7,124,205; 7,237,036; 7,337,241; 7,673,072; 8,131,880; 8,805,948; and 9,055,104 (collectively, the "Asserted Patents"). Alacritech is the legal owner by assignment of the Asserted Patents, which were all duly and legally issued by the United States Patent and Trademark Office. Alacritech seeks injunctive relief and monetary damages.

JURISDICTION AND VENUE

7. This is a civil action for patent infringement arising under the patent laws of the United States, 35 U.S.C. § 1, *et seq.*, including § 271. This Court has subject matter jurisdiction over this action under 28 U.S.C. §§ 1331 and 1338(a).

8. This Court has personal jurisdiction over Defendants because, among other reasons, Defendants regularly conducts business in Texas, including in this District, and Defendants have committed and continue to commit direct and indirect acts of patent infringement complained of herein within this District and elsewhere in Texas and the United States.

9. Wistron maintains a manufacturing center in McKinney, Texas and a service center in Dallas, Texas, and Wistron uses these facilities to build, repair, sell and/or provide a variety of products and services, including products and services complained of herein.

10. Wiwynn and Wistron have purposefully and voluntarily sold their infringing products and/or placed their infringing products into the stream of commerce (e.g., through their customers and their sales and service partners) with the expectation that those infringing products will be used in this District, and the infringing products have been and continue to be used in this District. Wiwynn and Wistron have also provided and continue to provide infringing services associated with their infringing products in this District.

11. SMS InfoComm is a Texas corporation that is based in and maintains a parts and service center in Grapevine, Texas, which it uses to build, repair, sell and/or provide a variety of products and services, including products and services complained of herein. SMS InfoComm sells and provides products and services, including infringing products and services complained of herein, directly to businesses in this District.

12. As such, Defendants have purposefully availed themselves of the privilege of conducting business within this District, have established sufficient minimum contacts with this District such that Defendants should reasonably and fairly anticipate being haled into this Court, and have purposefully directed activities at residents in this District, wherein at least a portion of the claims alleged herein arise out of or are related to those activities. Wistron has also participated in previous patent cases in this District.

13. Venue is proper in this District under 28 U.S.C. §§ 1391(b)-(c) and 1400(b) at least because, as discussed above, Defendants are subject to personal jurisdiction in this District,

regularly conduct business in this District, and have committed and continue to commit direct and indirect acts of patent infringement complained of herein within this District.

FACTUAL BACKGROUND

Alacritech's History

14. Alacritech was founded in 1997 by technology pioneer Larry Boucher, the creator and author of the original Small Computer System Interface ("SCSI") specification and a visionary, award-winning leader in server adapter, storage and networking technologies.

15. Mr. Boucher has more than 50 years of experience in the industry. Following a twelve-year tenure in IBM's Storage Division, he served as director of design services at Shugart Associates, where he developed the Shugart Associates System Interface ("SASI") and then SCSI, the industry standard for connecting storage and other peripherals to PCs and servers. In 1981, Mr. Boucher founded Adaptec, Inc., which became a global leader for host adapters and other innovative storage solutions. After taking Adaptec public, Mr. Boucher founded Auspex Systems, Inc., a manufacturer of enterprise servers, where he pioneered the networked file system design that is the basis of today's network-attached storage ("NAS") model. In 1997, Mr. Boucher founded Alacritech.

16. Mr. Boucher and other innovators at Alacritech (including Peter Craft, Clive Philbrick, Stephen Blightman, David Higgen, and Daryl Starr) foresaw the convergence of storage and networking and, as a result, the enormous processing demands that would be placed on host computer CPUs in order to move and store large quantities of data within a network, creating bottlenecks and reducing CPU processing power available for performing more substantive computing tasks. To solve this impending problem, the Alacritech team pioneered a series of fundamental network acceleration technologies, including but not limited to techniques for streamlining, bypassing and/or offloading aspects of conventional network protocol

processing from host CPUs to "intelligent" network interface devices (sometimes called "NIDs"). These technologies are critical to modern network computing, dramatically increasing the speed and efficiency with which data is transferred and stored, while reducing the associated processing burden imposed on host CPUs.

17. Working with industry partners, Alacritech released a number of network and storage products related to the technologies it developed. For example, Alacritech produced a series of Scalable Network Accelerators (also referred to as TCP Offload Engine (TOE) Network Interface Cards (TNICs)) and the ANX 1500, a sophisticated Network File System (NFS) Throughput Acceleration Appliance for use with Network-Attached Storage (NAS) systems.

Alacritech's Asserted Patents

18. Network computing is ubiquitous in contemporary society. It enables the dissemination of information and digital content to people around the world, and it is a critical component of the modern information economy. Many businesses have their own data centers (made up of large numbers of networked servers) that they use for the remote storage, management, processing, and/or distribution of their data. And a lucrative and rapidly-growing industry has developed to provide cloud computing services, which essentially allow businesses (and consumers) to offload their data to shared, third-party data centers.

19. However, as growing volumes of data are moved across networks of increasing complexity and bandwidth, more and more of the processing power of the servers (and/or other computers) in a network is consumed by simply moving and storing the data, greatly diminishing the ability of the servers to perform other more substantive tasks. In addition, bottlenecks develop when there is insufficient processing power available to transfer data, and the data cannot be moved as quickly or efficiently as desired. The Alacritech team foresaw these

problems years ago and, to address them, they developed and patented a collection of innovative network acceleration techniques that dramatically speed up the transfer and storage of data, and decrease the corresponding processing demands on servers. Alacritech holds 71 United States patents covering its groundbreaking inventions.

20. Conventionally, computers connected over a network rely on a multi-layered software architecture to transfer and store data. The architecture (also called a protocol stack) is generally based on one or more specifications and/or protocols, such as TCP/IP (a protocol suite including the Transmission Control Protocol ("TCP") and Internet Protocol ("IP")). Depending on the specifications and/or protocols at issue, the architecture may include up to seven different layers described by the Open Systems Interconnection (OSI) model (listed, in order, from highest to lowest): the application layer, the presentation layer, the session layer, the transport layer, the network layer, the data link layer, and the physical layer. Each software layer performs different functions associated with transferring and storing data.

21. In order to prepare data for transmission over a network, a sending computer must process the data through each layer of the protocol stack (working from highest to lowest). At each layer, the sending computer must perform further processing on the data resulting from processing by the previous layer, such as preparing and attaching a new header containing associated metadata. In the transport layer, the sending computer must also divide the data up into units (e.g., packets) that are small enough to be transmitted over the network medium. Likewise, a receiving computer must process incoming data through each layer of the same protocol stack (working from lowest to highest) before it can be used by host applications on the receiving computer. At each layer, the receiving computer must perform further processing on the data resulting from processing by the previous layer, such as removing and processing an

additional header and, in the transport layer, combining multiple units of data together in memory. In both the sending and receiving operations, the computer may move and/or make a new copy of the data each time another layer is processed.

22. This conventional approach for transferring and storing data within a network, while functional, is also slow and highly inefficient. Too much of the processing power of a computer's CPU is wasted performing brute-force, layer-by-layer processing. Recognizing these inefficiencies early on, the Alacritech team developed sophisticated solutions that greatly improve upon the conventional approach. Central to these solutions is Alacritech's pioneering use of dedicated NIDs to efficiently handle optimal portions of the processing associated with sending, receiving, and storing data, particularly for complicated multi-layer protocol suites that use variable-length units of data with multiple headers, such as TCP/IP.

23. Others in the industry attempted solutions that involved offloading essentially all processing associated with data communications to a NID, completely bypassing the host CPU, but they were unable to develop effective solutions that were suitable for TCP/IP and other more complicated protocol suites involving maintenance of state and variable-length data units with multiple headers, situations in which it is important that the host CPU remains involved in the processing. Still others in the industry attempted solutions involving offload of discrete TCP/IP processing tasks to a NID, but they were only able to offload very limited tasks, such as checksum processing, that did not significantly reduce the processing burden on the host CPU.

24. Only Alacritech solved the problem of how to efficiently offload large portions of communications processing to NIDs implementing TCP/IP and other more complicated protocol suites. By offloading processing tasks to a dedicated NID in accordance with the solutions provided by the Asserted Patents, transfer of data between devices is accelerated and the host

CPUs retain dramatically more processing power to perform other more substantive tasks, without sacrificing the flexibility and control necessary to implement a complicated protocol suite such as TCP/IP. Some of the innovative solutions and techniques developed by Alacritech and claimed by the Asserted Patents are discussed below.

25. Some of the Asserted Patents provide solutions that increase efficiency of network computing by using multiple paths to process received data, whereby an intelligent NID uses criteria (e.g., provided by the host CPU) to determine whether incoming data should be processed directly by the NID (e.g., using a "fast-path"), or whether it should be passed to and processed by the host CPU in the conventional manner (e.g., using a "slow-path"). This substantially reduces the burden on the host CPU, while retaining its flexibility and control for handling exceptions and other complicated processing tasks. And, because the NID is specifically designed to perform these operations (usually in hardware), it can perform them more efficiently and more quickly than a host CPU.

26. Alacritech also pioneered techniques that allow NIDs to transfer and store data more quickly and efficiently. For example, some of the Asserted Patents provide solutions through which NIDs transfer data to and from host memory associated with upper layer (e.g., application layer) software, without unit-by-unit lower-layer processing by the host CPU and without excessive intermediate copying of data in a series of intermediate buffers and/or caches. In this way, data can be transferred to the network or the host memory directly by the NID, without substantial processing by the host CPU or excessive intermediate copying, resulting in much faster and more efficient transfer and storage of data.

27. As another example, some of the Asserted Patents provide solutions for more efficiently preparing and sending data over a network. For example, the NID uses information

(e.g., provided by the host CPU) to divide data up into smaller units for transmission, and to generate and attach multiple lower-layer headers to the units of data. Moreover, it does so in essentially a single operation, without the brute-force, layer-by-layer processing (and associated copying) required to carry out the same tasks under the conventional approach described above. These solutions greatly reduce processing by the host CPU associated with sending data, and they allow the NID to prepare and transmit data more efficiently.

28. Similarly, some of the Asserted Patents provide solutions for more efficiently processing data received from a network. For example, the NID removes and processes the lower-layer headers from related units of data, without the brute-force processing of those headers required under the conventional approach described above. In addition, the NID may combine processed data units into a single payload of data that it delivers to host memory. These solutions greatly reduce processing by the host CPU associated with receiving and storing data, and they allow the NID to receive and process data more efficiently.

29. The Asserted Patents are a product of Alacritech's extensive research and development, reflecting groundbreaking innovations that are found throughout modern networks, especially large-scale and/or high-end networks such as those used in data centers.

30. The technologies covered by Alacritech's Asserted Patents are critical to fundamental network and/or storage acceleration techniques, including Large Segment Offload ("LSO") (also called Large Send Offload or Generic Segmentation Offload), Receive Side Coalescing ("RSC") (also called Receive Segment Coalescing, Large Receive Offload, or Generic Receive Offload), and TCP Offload Engine ("TOE"). They are also essential to many network and/or storage protocols that incorporate Remote Direct Memory Access ("RDMA"), such as the InfiniBand protocol, the RDMA over Converged Ethernet ("RoCE") protocol, the

Internet Wide Area RDMA Protocol ("iWARP"), the Internet Small Computer System Interface ("iSCSI") Extensions for RDMA ("iSER") protocol, and the Server Message Block ("SMB") Direct protocol.

31. Without the benefit of Alacritech's groundbreaking inventions, modern computing networks (especially large-scale and/or high-performance networks) would be significantly slower and less efficient (and, therefore, more expensive and less useful) than they are today.

Defendants' Infringing Technologies

32. Defendants use Alacritech's patented technologies across many different parts of their businesses, including in the data centers they help their customers build and operate, in the servers and other network products that they make, use and/or sell, and in the associated consulting and support services that they provide to their customers. Defendants infringe the Asserted Patents directly, and also induce and contribute to infringement of the Asserted Patents by their customers.

33. Wistron makes, uses and/or sells a variety of network products that infringe the Asserted Patents, including servers, storage devices, network adapters and other network interface devices, and converged products, which Wistron's customers sell under their own brand names. Wistron also provides a range of services, through which Wistron designs, builds and/or provides technical support for infringing networking infrastructure for its customers, often using the infringing network products described above. *See, e.g.*, "About Wistron," available at http://www.wistron.com/about/about_wistron.htm; "Wistron Products and Services," available at http://www.wistron.com/about/products_and_services.htm; "Wistron Products," available at http://www.wistron.com/about/products_and_services_products.htm.

34. Wiwynn makes, uses and/or sells a variety of servers, storage devices, and other network products (including products manufactured by Wistron) that infringe the Asserted Patents. In addition, Wiwynn provides a large portfolio of network solutions and services, including end-to-end Data Center and Cloud solutions and other technical support services, through which Wiwynn designs, builds and/or provides technical support for infringing data centers and other infringing networking infrastructure for its customers, often using the infringing network products described above. *See, e.g.,* "Wiwynn Company Overview," available at http://www.wiwynn.com/english/page/index/3; "Wiwynn Products & Services," available at http://www.wiwynn.com/english/product; "Wiwynn Service & Integration," available at http://www.wiwynn.com/english/product/service; "Wistron Group," available at http://www.wistron.com/about/wistron_group.htm.

35. SMS InfoComm sells parts and provides its customers with technical support and repair services for a variety network products (including products from Wiwynn and Wistron) that infringe the Asserted Patents, including servers, storage devices, cloud computing devices, and other network products. *See, e.g.,* "SMS InfoComm Corporation Service," available at http://www.smsinfocomm.com/Files/index_Service.html.

36. A large and growing portion of Defendants' revenue is tied to the infringing network products and services described above. Without the benefit Alacritech's patented technologies, the infringing network products and services Defendants provide would suffer a significant degradation in performance, causing a substantial decrease in Defendants' revenue.

COUNT I

INFRINGEMENT OF U.S. PATENT NO. 7,124,205

37. Alacritech re-alleges and incorporates by reference each of the allegations of the paragraphs set forth above as though fully set forth herein.

38. Alacritech is the current exclusive owner and assignee of all right, title, and interest in and to U.S. Patent No. 7,124,205 (the "'205 patent"), titled "Network Interface Device that Fast-Path Processes Solicited Session Layer Read Commands," duly and legally issued by the United States Patent and Trademark Office on October 17, 2006, including the right to bring this suit for injunctive relief and damages. A true and correct copy of the '205 patent is attached hereto as Exhibit A.

39. The '205 patent is valid and enforceable.

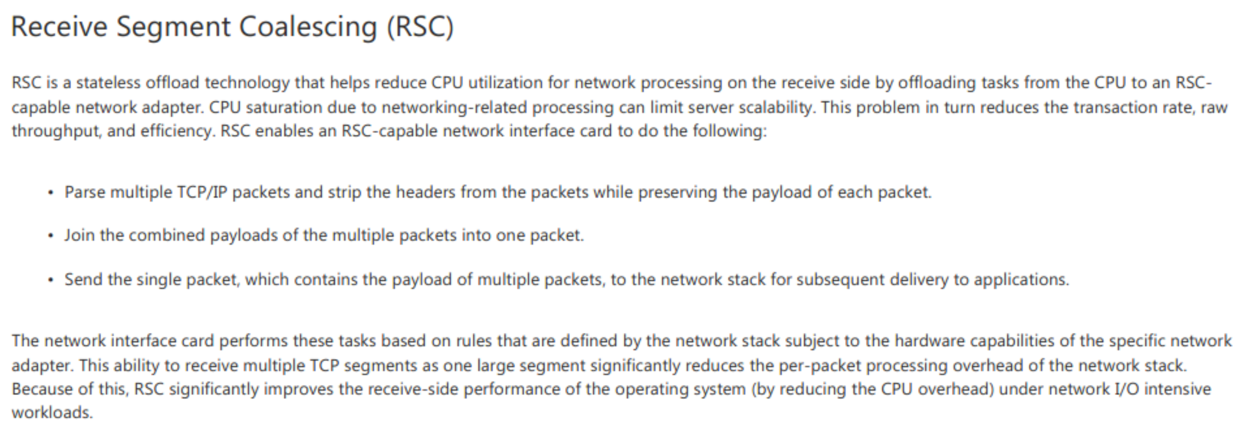
40. Defendants have directly infringed and are currently directly infringing the '205 patent by making, using, selling, offering for sale, and/or importing into the United States, without authority, products, methods, equipment, and/or services that practice one or more claims of the '205 patent in connection with infringing RSC functionality, including but not limited to the Wiwynn SV100G2, SV300, SV300G2, SV320, SV320G2, SV324G2, SV5270G2-R, SV5270G2-S, SV7110, SV7220G2-N, SV7220G2-P, SV7220G2-S, SV7220G2-V, NM10GR, and NM10GS; any other activities, products and/or services involving the products identified above; and any other activities, products and/or services that practice and/or support similarly infringing RSC functionality (collectively, "the '205 Accused Products"). The '205 Accused Products are non-limiting examples that were identified based on publicly available information, and Alacritech reserves the right to identify additional infringing activities, products and services, including, for example, on the basis of information obtained during discovery.

41. As just one non-limiting example, set forth below (with claim language in italics) is a description of Defendants' infringement of exemplary claim 5 (including claim 1, upon which claim 5 depends) of the '205 patent in connection with the Wiwynn SV7220G2 Server configured with a Wiwynn NM10GR Mezzanine Card. This description is based on publicly

available information. Alacritech reserves the right to modify this description, including, for example, on the basis of information about the '205 Accused Products that it obtains during discovery.

1(a) *An apparatus comprising: a host computer having a protocol stack and a destination memory, the protocol stack including a session layer portion, the session layer portion being for processing a session layer protocol;* – Defendants make, use, sell, offer for sale, and/or import servers, network interface devices (e.g., controllers and cards), and other network products that support infringing RSC functionality. One example of an infringing products is the Wiwynn SV7220G2 Server (the "SV7220G2 Server") configured with a Wiwynn NM10GR Mezzanine Card (the "NM10GR Card"). *See, e.g.*, "Wiwynn 2016 OCP US Summit Product Brochure," available at https://www.circleb.eu/wp-content/uploads/2016/04/OCP-US-Summit-OPS.pdf ("OCP Product Brochure"); "Wiwynn SV7220G2 Series Datasheet," available at http://www.wiwynn.com/usr_files/20150413140223_Datasheet%20-%20SV7220G2_150413.pdf ("SV7220G2 Datasheet"). The SV7220G2 Server comprises a host computer that has destination memory and a protocol stack (e.g., of the server's Microsoft Windows or Linux operating system) including a session layer portion for processing a session layer protocol, such as iSCSI. *See, e.g.*, OCP Product Brochure; SV7220G2 Datasheet; "Introduction of iSCSI Target in Windows Server 2012" available at https://blogs.technet.microsoft.com/filecab/2012/05/21/introduction-of-iscsi-target-in-windows-server-2012. Microsoft describes how using a network interface device supporting infringing RSC functionality reduces processing overhead of the protocol stack on a server (such as the SV7220G2 Server incorporating the NM10GR Card) running the Microsoft Windows Server

Operating System, which enables RSC by default. *See, e.g.*, "Receive Segment Coalescing," available at https://technet.microsoft.com/en-us/library/hh997024.aspx.

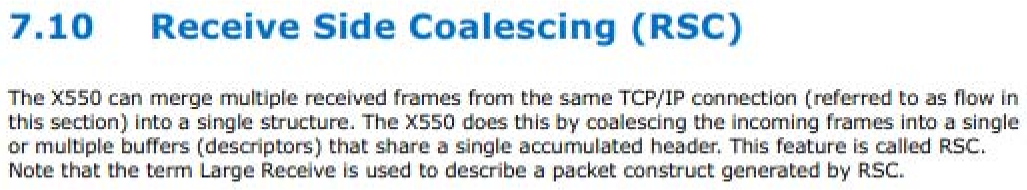

Receive Segment Coalescing (RSC)

RSC is a stateless offload technology that helps reduce CPU utilization for network processing on the receive side by offloading tasks from the CPU to an RSC-capable network adapter. CPU saturation due to networking-related processing can limit server scalability. This problem in turn reduces the transaction rate, raw throughput, and efficiency. RSC enables an RSC-capable network interface card to do the following:

- Parse multiple TCP/IP packets and strip the headers from the packets while preserving the payload of each packet.
- Join the combined payloads of the multiple packets into one packet.
- Send the single packet, which contains the payload of multiple packets, to the network stack for subsequent delivery to applications.

The network interface card performs these tasks based on rules that are defined by the network stack subject to the hardware capabilities of the specific network adapter. This ability to receive multiple TCP segments as one large segment significantly reduces the per-packet processing overhead of the network stack. Because of this, RSC significantly improves the receive-side performance of the operating system (by reducing the CPU overhead) under network I/O intensive workloads.

1(b) *and a network interface device coupled to the host computer,* – The SV7220G2 Server incorporating the NM10GR Card includes a network interface device coupled to the host computer, comprising the NM10GR Card with its Intel X550 controller (the "X550 Controller"). *See, e.g.*, OCP Product Brochure; SV7220G2 Datasheet. The X550 Controller provides "unified networking delivering . . . iSCSI." *See, e.g.*, "Intel Ethernet Controller X550 Product Brief" ("X550 Product Brief"), available at http://www.intel.com/content/www/us/en/embedded/products/networking/ethernet-x550-brief.html. An introduction to the X550 Controller's RSC functionality is provided in Section 7.10 of the "Intel Ethernet Controller X550 Datasheet," Revision 2.1 (May 2016), available at http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/ethernet-x550-datasheet.pdf (the "X550 Datasheet").


7.10 Receive Side Coalescing (RSC)

The X550 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X550 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

1(c) *the network interface device receiving from outside the apparatus a response to a solicited read command, the solicited read command being of the session layer protocol,* – The network interface device of a SV7220G2 Server incorporating the NM10GR Card receives responses to solicited read commands of the session layer protocol issued by the SV7220G2 Server over networks that are outside the apparatus. This is indicated, for example, in § 1.3.2 ("Network Interfaces") at 21 of the X550 Datasheet. *See, also, e.g.*, Internet Engineering Task Force (IETF), Request for Comments: 7143 "Internet Small Computer System Interface (iSCSI) Protocol (Consolidated)," available at http://www.rfc-base.org/txt/rfc-7143.txt, at Appendix A at 253 ("Read Operation Example").

1(d) *performing fast-path processing on the response such that a data portion of the response is placed into the destination memory without the protocol stack of the host computer performing any network layer processing or any transport layer processing on the response.* – In connection with the RSC functionality discussed above, the network interface device of a SV7220G2 Server incorporating the NM10GR Card performs fast-path processing on a response such that a data portion of the response is placed into destination memory without the protocol stack of the SV7220G2 Server performing any network layer processing or any transport layer processing on the response. This is described in Section 7.10 of the X550 Datasheet, excerpts of which are provided below.

7.10 Receive Side Coalescing (RSC)

The X550 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X550 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

The X550 digests received packets and categorizes them by their TCP/IP connections (flows). For each flow, hardware coalesces the packets as shown in Figure 7-41 and Figure 7-42 (the colored parameters are explained in the RSC context table and receive descriptor sections). The X550 can handle up to 32 concurrent flows per LAN port at any given time. Each flow handled by RSC offload has an associated context. The X550 opens and closes the RSC contexts autonomously with no need for any software intervention. Software needs only to enable RSC in the selected receive queues.

***

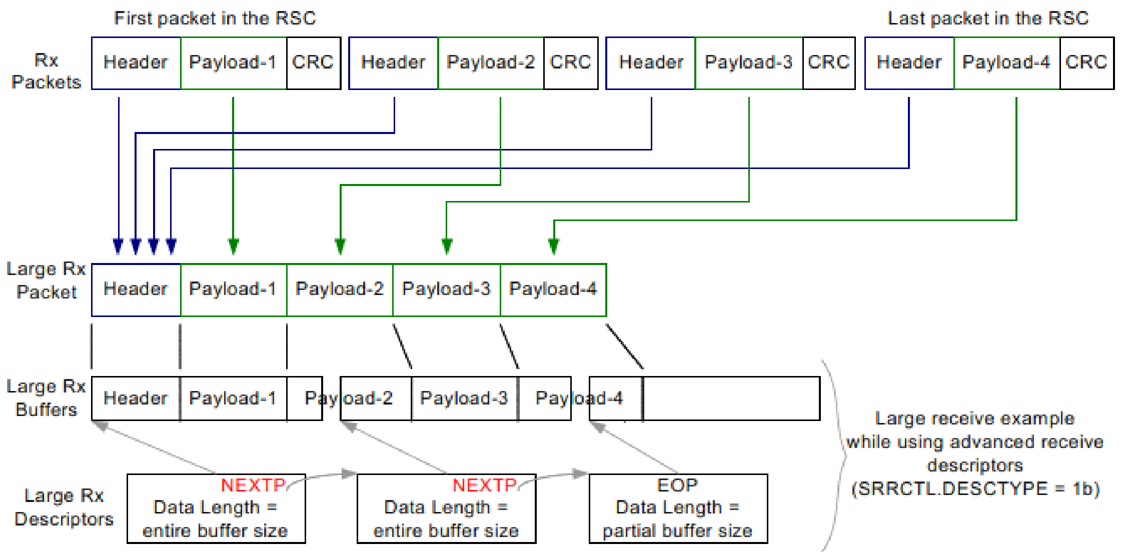


Figure 7-41 RSC Functionality (No Header Split)

***

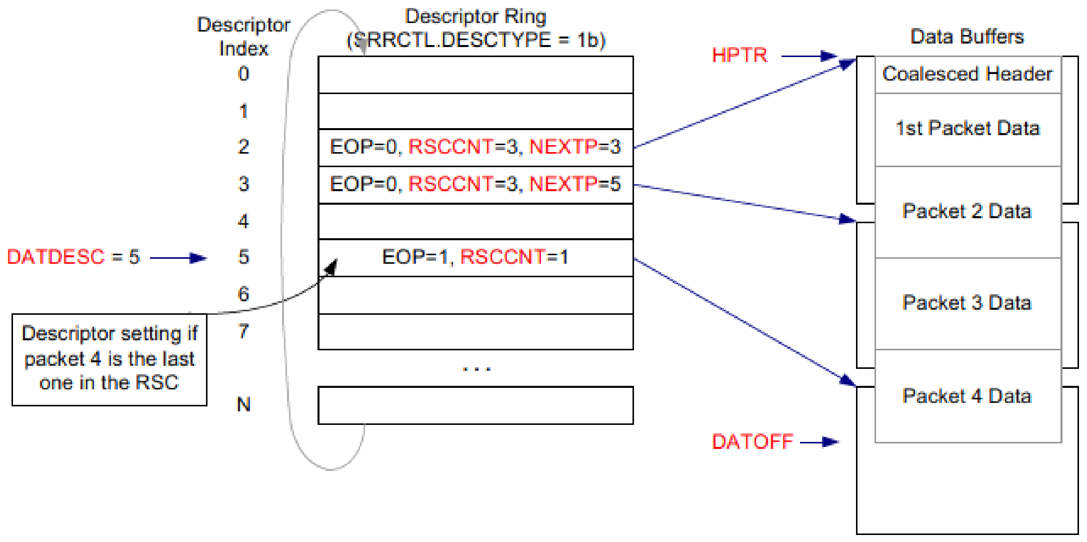


Figure 7-42 RSC Functionality (No Header Split)

5. *The apparatus of claim 1, wherein the protocol stack of the host computer can process a second response to a second solicited read command of the session layer protocol, the protocol stack processing the second response such that the protocol stack performs both network layer processing and transport layer processing on the response.* – In connection with the RSC functionality discussed above, the SV7220G2 Server incorporating the NM10GR Card can process a second response to a second solicited read command of the session layer protocol (e.g., iSCSI protocol) such that the protocol stack of the SV7220G2 Server performs both network layer and transport layer processing on the response. As described in Section 7.10.1 and shown in Figure 7-43 (and elsewhere) in the X550 Datasheet, if a response does not qualify for fast-path processing, it is sent to the host protocol stack for conventional processing.

7.10.1 Packet Candidacy for RSC

Incoming packets can be good candidates for RSC offload when the following conditions are met. If any of the these conditions are not met, and assuming the queue is configured as required in Section 4.6.7.2 the received packet is processed in the legacy (non-coalescing) scheme.

***

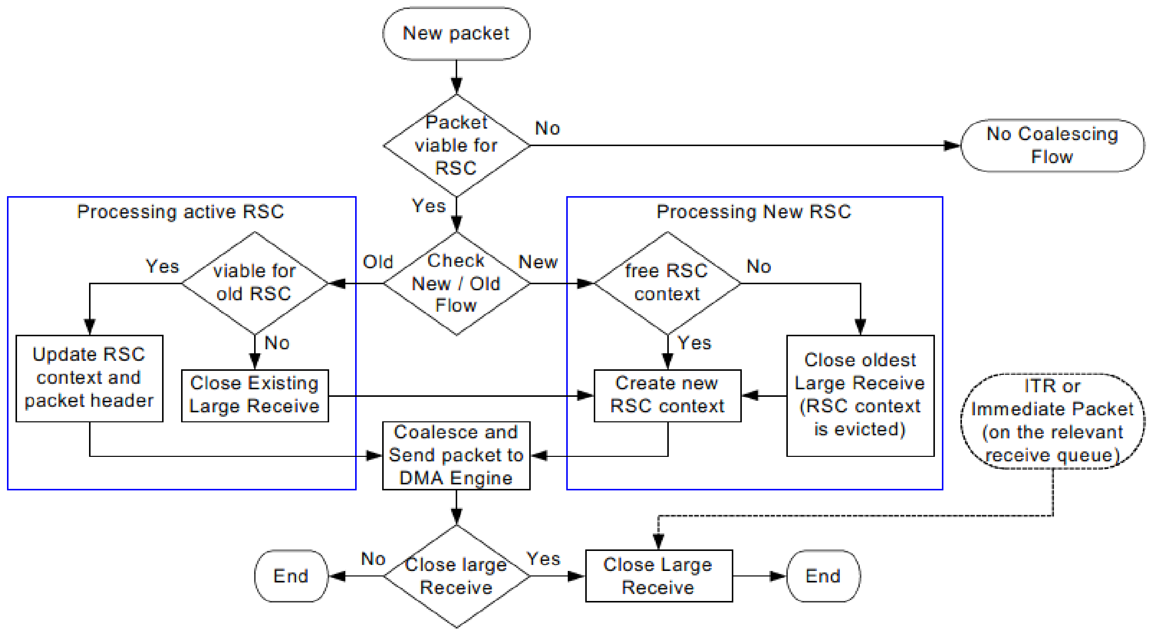


Figure 7-43 RSC Event Flow

42. As of no later than the filing and service of this Complaint, Defendants are also indirectly infringing the '205 patent.

43. Defendants have actual knowledge of Alacritech's rights in the '205 patent and details of Defendants' infringement of the '205 patent based on at least the filing and service of this Complaint.

44. Defendants make, use, import, offer for sale, and/or sell the '205 Accused Products (including, for example, the products described above) with knowledge of or willful blindness to the fact that their actions will induce their customers to infringe the '205 patent. Defendants induce others to infringe the '205 patent in violation of 35 U.S.C. § 271 by encouraging and facilitating others to practice the '205 patent's inventions for accelerated network communications with intent that those performing the acts infringe the '205 patent. For example, Defendants sell and/or provide '205 Accused Products that perform infringing RSC functionality by default. When Defendants' customers use the '205 Accused Products for their intended purpose, they practice one or more claims of the '205 patent. Defendants instructs their

customers how to set up and use '205 Accused Products. In some cases, Defendants even sets up '205 Accused Products for their customers. Once they have been set up, Defendants' customers infringe one or more claims of the '205 patent simply by using the '205 Accused Products. By selling and/or providing their customers '205 Accused Products that infringe by default when used for their intended purpose, and by setting up for their customers and/or instructing their customers how to set up and use those products, Defendants induce those customers to infringe the '205 patent.

45. Defendants also contribute to the infringement of the '205 patent in violation of 35 U.S.C. § 271. Defendants know that infringing components of the '205 Accused Products are especially made or especially adapted for use in the infringement of the '205 patent. The infringing components of these products are not staple articles or commodities of commerce suitable for substantial non-infringing use, and the infringing components of these products are a material part of the invention of the '205 patent. The '205 Accused Products contain infringing components such as specialized hardware and/or software for performing infringing RSC functionality, which are separable from the '205 Accused Products, material to practicing the '205 patent's inventions for accelerated network communications, and have no substantial non-infringing use. Accordingly, Defendants are also contributing to the direct infringement of the '205 patent by their customers.

46. Defendants are not licensed or otherwise authorized by Alacritech to practice, contributorily practice and/or induce third parties to practice the claims of the '205 patent.

47. By reason of Defendants' infringing activities, Alacritech has suffered, and will continue to suffer, substantial damages in an amount to be proven at trial. Alacritech is entitled to a money judgment in an amount adequate to compensate for Defendants' infringement, but in

no event less than a reasonable royalty for the use made of the invention by Defendants, together with interest and costs as fixed by the Court.

48. Defendants' continuing acts of infringement have caused and will continue to cause Alacritech irreparable harm, for which Alacritech has no adequate remedy at law, unless Defendants' continuing acts of infringement are enjoined by the Court. The hardships that an injunction would impose are less than those faced by Alacritech should an injunction not issue. The public interest would be served by issuance of an injunction.

49. Defendants' infringement of the '205 patent is exceptional and entitles Alacritech to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

COUNT II

INFRINGEMENT OF U.S. PATENT NO. 7,237,036

50. Alacritech re-alleges and incorporates by reference each of the allegations of paragraphs 1-36 set forth above as though fully set forth herein.

51. Alacritech is the current exclusive owner and assignee of all right, title, and interest in and to U.S. Patent No. 7,237,036 (the "'036 patent"), titled "Fast-Path Apparatus for Receiving Data Corresponding a TCP Connection," duly and legally issued by the United States Patent and Trademark Office on June 26, 2007, including the right to bring this suit for injunctive relief and damages. A true and correct copy of the '036 patent is attached hereto as Exhibit B.

52. The '036 patent is valid and enforceable.

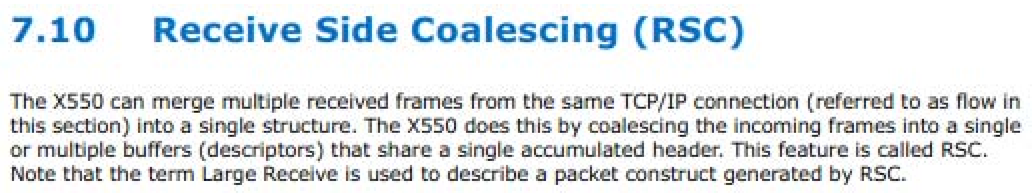
53. Defendants have directly infringed and are currently directly infringing the '205 patent by making, using, selling, offering for sale, and/or importing into the United States, without authority, products, methods, equipment, and/or services that practice one or more claims of the '036 patent in connection with infringing RSC functionality, including but not

limited to the Wiwynn SV100G2, SV300, SV300G2, SV320, SV320G2, SV324G2, SV5270G2-R, SV5270G2-S, SV7110, SV7220G2-N, SV7220G2-P, SV7220G2-S, SV7220G2-V, NM10GR, and NM10GS; any other activities, products and/or services involving the products identified above; and any other activities, products and/or services that practice and/or support similarly infringing RSC functionality (collectively, "the '036 Accused Products"). The '036 Accused Products are non-limiting examples that were identified based on publicly available information, and Alacritech reserves the right to identify additional infringing activities, products and services, including, for example, on the basis of information obtained during discovery.

54. As just one non-limiting example, set forth below (with claim language in italics) is a description of Defendants' infringement of exemplary claim 1 of the '036 patent in connection with the Wiwynn SV7220G2 Server configured with a Wiwynn NM10GR Mezzanine Card. This description is based on publicly available information. Alacritech reserves the right to modify this description, including, for example, on the basis of information about the '036 Accused Products that it obtains during discovery.

1(a) *A device for use with a first apparatus that is connectable to a second apparatus, the first apparatus containing a memory and a first processor operating a stack of protocol processing layers that create a context for communication, the context including a media access control (MAC) layer address, an Internet Protocol (IP) address and Transmission Control Protocol (TCP) state information, the device comprising:* – Defendants make, use, sell, offer for sale, and/or import servers, network interface devices (e.g., controllers and cards), and other network products that support infringing RSC functionality. One example of an infringing products is the Wiwynn SV7220G2 Server (the "SV7220G2 Server") configured with a Wiwynn NM10GR Mezzanine Card (the "NM10GR Card"). To the extent the preamble is limiting, the

SV7220G2 Server incorporating the NM10GR Card includes a device for use with a first apparatus, comprising at least the NM10GR Card with its Intel X550 controller (the "X550 Controller"). *See, e.g.*, "Wiwynn SV7220G2 Series Datasheet," available at http://www.wiwynn.com/usr_files/20150413140223_Datasheet%20-%20SV7220G2_150413.pdf ("SV7220G2 Datasheet"); "Wiwynn 2016 OCP US Summit Product Brochure," available at https://www.circleb.eu/wp-content/uploads/2016/04/OCP-US-Summit-OPS.pdf ("OCP Product Brochure"). A SV7220G2 Server comprises a first apparatus that is connectable to a second apparatus (e.g., another server), and contains memory and a first processor (e.g., an Intel Xeon processor). *See, e.g.,* SV7220G2 Datasheet. The first processor on the server operates a stack of protocol processing layers (e.g., the protocol stack of the server's Microsoft Windows or Linux operating system) that creates a context for communication that includes a MAC address, an IP address, and TCP state information for the communication. *See, e.g.,* SV7220G2 Datasheet. The "Intel Ethernet Controller X550 Datasheet," Revision 2.1 (May 2016), available at http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/ethernet-x550-datasheet.pdf (the "X550 Datasheet") provides an overview of the X550 Controller and explains how it interacts with a compatible server (such as the SV7220G2 Server), including, for example, describing in Section 7.10 how it performs RSC functionality associated with processing incoming data packets. Section 7.10.1 of the X550 Datasheet describes the structure of a packet, which contains information (e.g., a MAC address and an IP address) corresponding to the context for communication to which the packet pertains.



7.10 Receive Side Coalescing (RSC)

The X550 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X550 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

***

7.10.1 Packet Candidacy for RSC

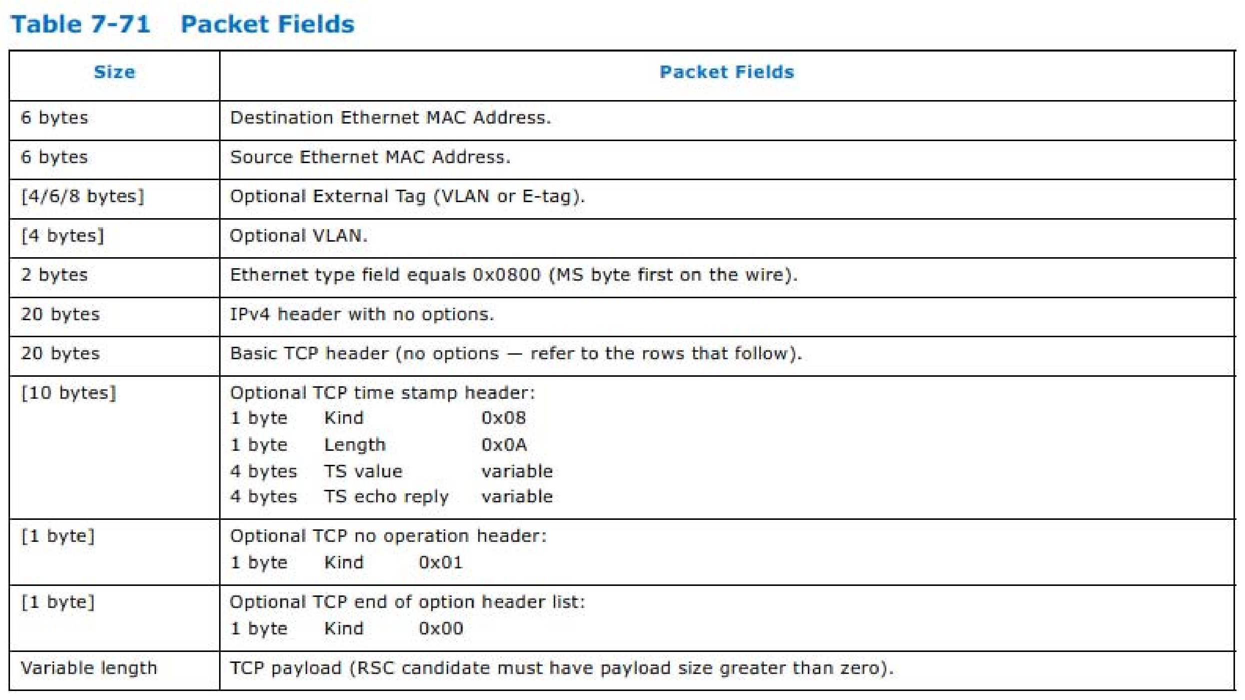
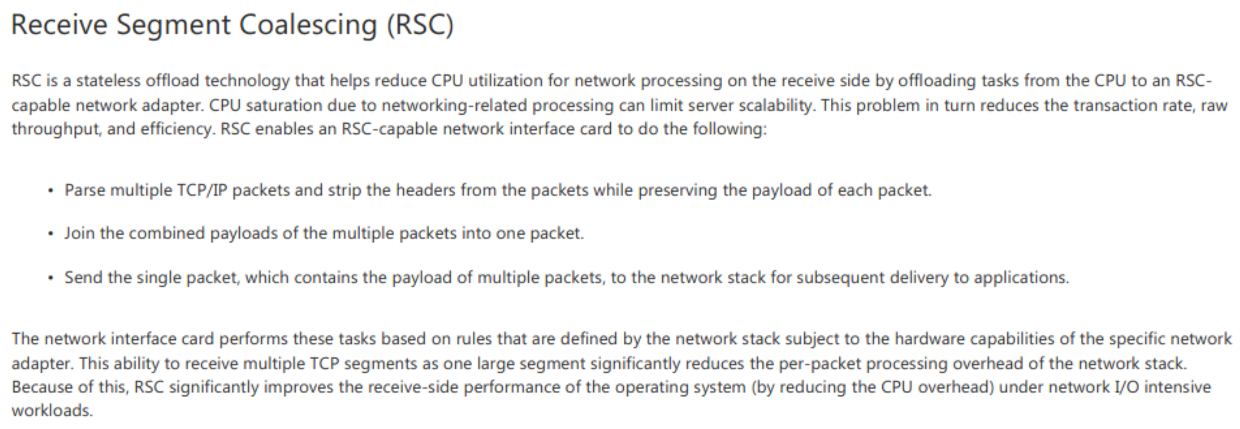


Table 7-71 Packet Fields

Size	Packet Fields
6 bytes	Destination Ethernet MAC Address.
6 bytes	Source Ethernet MAC Address.
[4/6/8 bytes]	Optional External Tag (VLAN or E-tag).
[4 bytes]	Optional VLAN.
2 bytes	Ethernet type field equals 0x0800 (MS byte first on the wire).
20 bytes	IPv4 header with no options.
20 bytes	Basic TCP header (no options — refer to the rows that follow).
[10 bytes]	Optional TCP time stamp header: 1 byte Kind 0x08 1 byte Length 0x0A 4 bytes TS value variable 4 bytes TS echo reply variable
[1 byte]	Optional TCP no operation header: 1 byte Kind 0x01
[1 byte]	Optional TCP end of option header list: 1 byte Kind 0x00
Variable length	TCP payload (RSC candidate must have payload size greater than zero).

Microsoft describes how a interface device supporting infringing RSC functionality reduces communications processing by the network protocol stack on a server (such as the SV7220G2 Server incorporating the NM10GR Card) running the Microsoft Windows Server Operating System, which enables RSC by default. *See, e.g.,* "Receive Segment Coalescing," available at https://technet.microsoft.com/en-us/library/hh997024.aspx.


Receive Segment Coalescing (RSC)

RSC is a stateless offload technology that helps reduce CPU utilization for network processing on the receive side by offloading tasks from the CPU to an RSC-capable network adapter. CPU saturation due to networking-related processing can limit server scalability. This problem in turn reduces the transaction rate, raw throughput, and efficiency. RSC enables an RSC-capable network interface card to do the following:

- Parse multiple TCP/IP packets and strip the headers from the packets while preserving the payload of each packet.
- Join the combined payloads of the multiple packets into one packet.
- Send the single packet, which contains the payload of multiple packets, to the network stack for subsequent delivery to applications.

The network interface card performs these tasks based on rules that are defined by the network stack subject to the hardware capabilities of the specific network adapter. This ability to receive multiple TCP segments as one large segment significantly reduces the per-packet processing overhead of the network stack. Because of this, RSC significantly improves the receive-side performance of the operating system (by reducing the CPU overhead) under network I/O intensive workloads.

1(b) *a communication processing mechanism connected to the first processor, said communication processing mechanism containing a second processor running instructions to process a message packet such that the context is employed to transfer data contained in said packet to the first apparatus memory* – The device comprises a communications processing mechanism connected to the server's processor and containing a second processor on the NM10GR Card running instructions to process an incoming message packet such that the context is used to transfer data in the packet to the server's memory using RSC functionality, as explained in Section 7.10 of the X550 Datasheet. For example, the second processor runs instructions using the context (e.g., its IP address information) to process an RSC-viable message packet, including removing and processing its headers and writing its data payload to the server's memory together with data from other packets corresponding to the same context, wherein the context is used to transfer the data to destination memory. The X550 Datasheet describes this functionality in the introduction to Section 7.10, portions of which are provided below.

7.10 Receive Side Coalescing (RSC)

The X550 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X550 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

The X550 digests received packets and categorizes them by their TCP/IP connections (flows). For each flow, hardware coalesces the packets as shown in Figure 7-41 and Figure 7-42 (the colored parameters are explained in the RSC context table and receive descriptor sections). The X550 can handle up to 32 concurrent flows per LAN port at any given time. Each flow handled by RSC offload has an associated context. The X550 opens and closes the RSC contexts autonomously with no need for any software intervention. Software needs only to enable RSC in the selected receive queues.

***

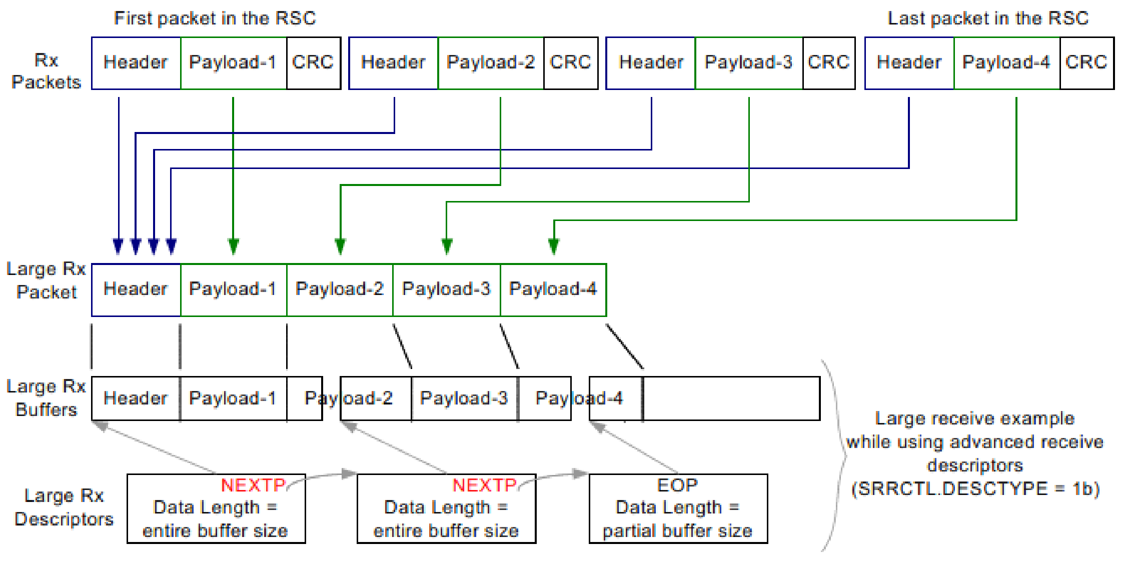


Figure 7-41 RSC Functionality (No Header Split)

***

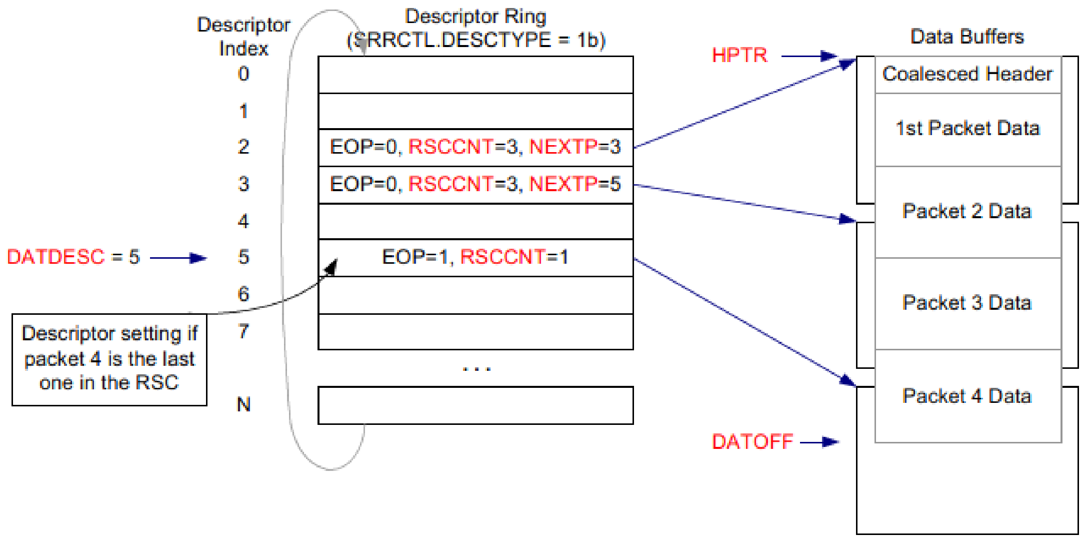


Figure 7-42 RSC Functionality (No Header Split)

1(c) *and the TCP state information is updated by said second processor.* – In connection with processing a message packet using the RSC functionality described above, the second processor of the NM10GR Card of a SV7220G2 Server updates the TCP state information, for example, by posting a coalesced header with updated TCP state information. The X550 Datasheet describes this functionality, for example, in Section 7.11.5.3, and the contents of the posted coalesced header are described in Section 7.11.1.

7.10.5.3 RSC Header

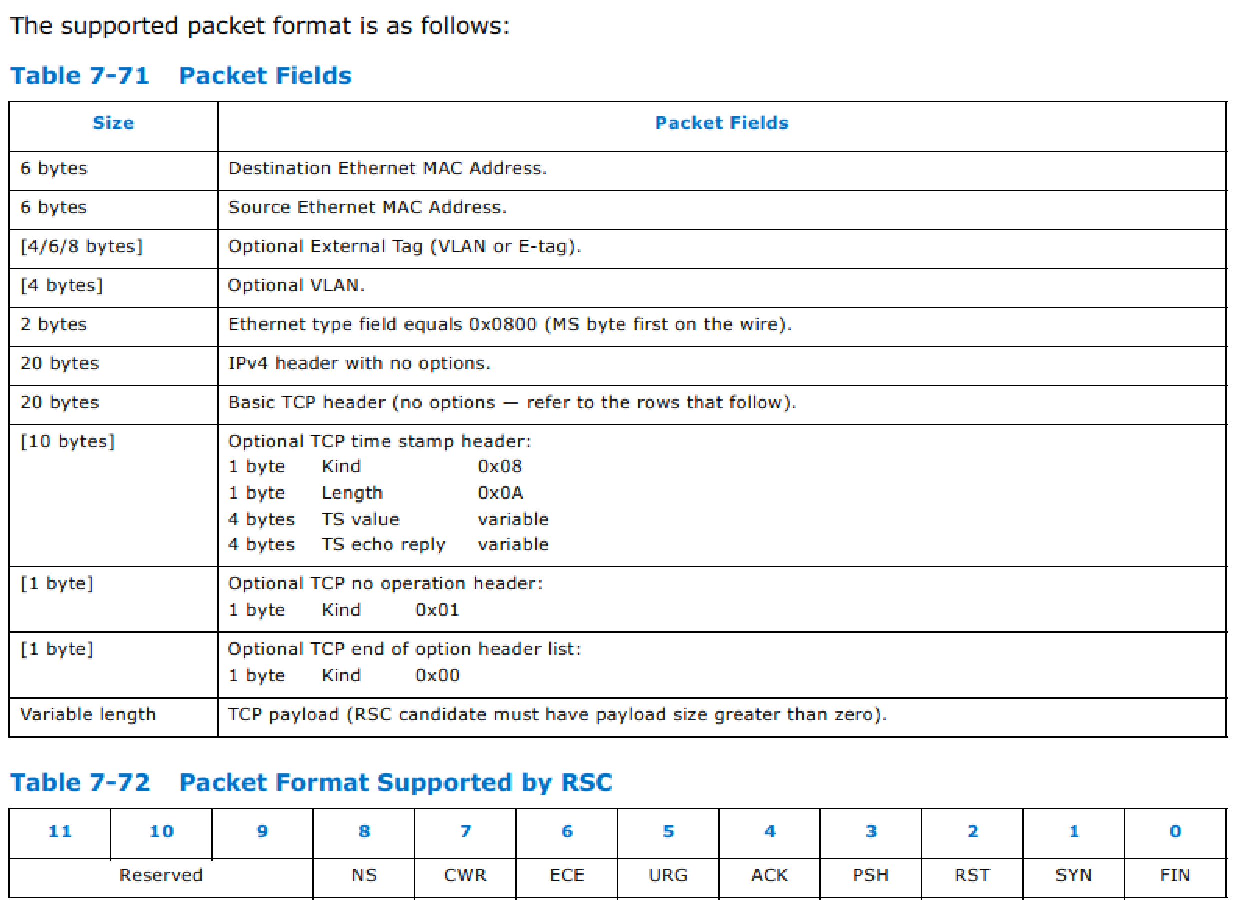
The RSC header is stored at the beginning of the first buffer when using advanced receive descriptors, or at the header buffer of the first descriptor when using header split descriptors (it is defined by the internal HPTR parameter in the RSC context). See Figure 7-44 for more details.

The packet's header is posted to host memory after it is updated by the RSC context as follows:

- **Packets with payload coalescing (RSCACKTYPE=0)** — The TCP sequence number is taken from the TCP context (it is taken from the first packet). The *Total Length* field in the IP header is taken from the RSC context (it represent the length of all coalesced packets). The IP checksum is re-calculated. The TCP checksum is set to zero.

***

7.10.1 Packet Candidacy for RSC



The supported packet format is as follows:

Table 7-71 Packet Fields

Size	Packet Fields
6 bytes	Destination Ethernet MAC Address.
6 bytes	Source Ethernet MAC Address.
[4/6/8 bytes]	Optional External Tag (VLAN or E-tag).
[4 bytes]	Optional VLAN.
2 bytes	Ethernet type field equals 0x0800 (MS byte first on the wire).
20 bytes	IPv4 header with no options.
20 bytes	Basic TCP header (no options — refer to the rows that follow).
[10 bytes]	Optional TCP time stamp header: 1 byte Kind 0x08 1 byte Length 0x0A 4 bytes TS value variable 4 bytes TS echo reply variable
[1 byte]	Optional TCP no operation header: 1 byte Kind 0x01
[1 byte]	Optional TCP end of option header list: 1 byte Kind 0x00
Variable length	TCP payload (RSC candidate must have payload size greater than zero).

Table 7-72 Packet Format Supported by RSC

11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN

55. As of no later than the filing and service of this Complaint, Defendants are also indirectly infringing the '036 patent.

56. Defendants have actual knowledge of Alacritech's rights in the '036 patent and details of Defendants' infringement of the '036 patent based on at least the filing and service of this Complaint.

57. Defendants make, use, import, offer for sale, and/or sell the '036 Accused Products (including, for example, the products described above) with knowledge of or willful blindness to the fact that their actions will induce their customers to infringe the '036 patent. Defendants induce others to infringe the '036 patent in violation of 35 U.S.C. § 271 by encouraging and facilitating others to practice the '036 patent's inventions for accelerated network communications with intent that those performing the acts infringe the '036 patent. For

example, Defendants sell and/or provide '036 Accused Products that perform infringing RSC functionality by default. When Defendants' customers use the '036 Accused Products for their intended purpose, they practice one or more claims of the '036 patent. Defendants instructs their customers how to set up and use '036 Accused Products. In some cases, Defendants even sets up '036 Accused Products for their customers. Once they have been set up, Defendants' customers infringe one or more claims of the '036 patent simply by using the '036 Accused Products. By selling and/or providing their customers '036 Accused Products that infringe by default when used for their intended purpose, and by setting up for their customers and/or instructing their customers how to set up and use those products, Defendants induce those customers to infringe the '036 patent.

58. Defendants also contribute to the infringement of the '036 patent in violation of 35 U.S.C. § 271. Defendants know that infringing components of the '036 Accused Products are especially made or especially adapted for use in the infringement of the '036 patent. The infringing components of these products are not staple articles or commodities of commerce suitable for substantial non-infringing use, and the infringing components of these products are a material part of the invention of the '036 patent. The '036 Accused Products contain infringing components such as specialized hardware and/or software for performing infringing RSC functionality, which are separable from the '036 Accused Products, material to practicing the '036 patent's inventions for accelerated network communications, and have no substantial non-infringing use. Accordingly, Defendants are also contributing to the direct infringement of the '036 patent by their customers.

59. Defendants are not licensed or otherwise authorized by Alacritech to practice, contributorily practice and/or induce third parties to practice the claims of the '036 patent.

60. By reason of Defendants' infringing activities, Alacritech has suffered, and will continue to suffer, substantial damages in an amount to be proven at trial. Alacritech is entitled to a money judgment in an amount adequate to compensate for Defendants' infringement, but in no event less than a reasonable royalty for the use made of the invention by Defendants, together with interest and costs as fixed by the Court.

61. Defendants' continuing acts of infringement have caused and will continue to cause Alacritech irreparable harm, for which Alacritech has no adequate remedy at law, unless Defendants' continuing acts of infringement are enjoined by the Court. The hardships that an injunction would impose are less than those faced by Alacritech should an injunction not issue. The public interest would be served by issuance of an injunction.

62. Defendants' infringement of the '036 patent is exceptional and entitles Alacritech to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

COUNT III

INFRINGEMENT OF U.S. PATENT NO. 7,337,241

63. Alacritech re-alleges and incorporates by reference each of the allegations of paragraphs 1-36 set forth above as though fully set forth herein.

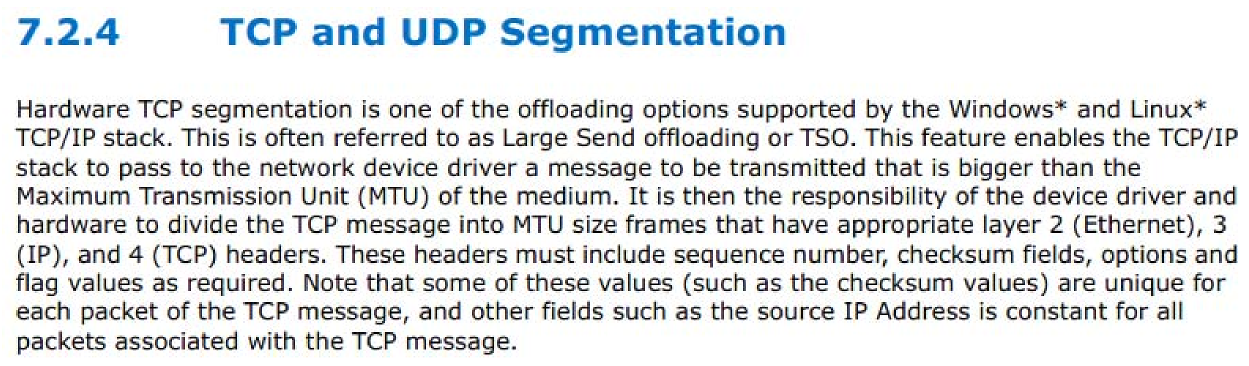
64. Alacritech is the current exclusive owner and assignee of all right, title, and interest in and to U.S. Patent No. 7,337,241 (the "'241 patent"), titled "Fast-Path Apparatus for Receiving Data Corresponding to a TCP Connection," duly and legally issued by the United States Patent and Trademark Office on February 26, 2008, including the right to bring this suit for injunctive relief and damages. A true and correct copy of the '241 patent is attached hereto as Exhibit C.

65. The '241 patent is valid and enforceable.

66. Defendants have directly infringed and are currently directly infringing the '241 patent by making, using, selling, offering for sale, and/or importing into the United States, without authority, products, methods, equipment, and/or services that practice one or more claims of the '241 patent in connection with infringing RSC and/or LSO functionality, including but not limited to the Wiwynn SV100G2, SV300, SV300G2, SV320, SV320G2, SV324G2, SV5270G2-R, SV5270G2-S, SV7110, SV7220-2A, SV7220-2P, SV7220-2S, SV7220G2-N, SV7220G2-P, SV7220G2-S, SV7220G2-V, NM10GR, and NM10GS; any other activities, products and/or services involving the products identified above; and any other activities, products and/or services that practice and/or support similarly infringing RSC and/or LSO functionality (collectively, "the '241 Accused Products"). The '241 Accused Products are non-limiting examples that were identified based on publicly available information, and Alacritech reserves the right to identify additional infringing activities, products and services, including, for example, on the basis of information obtained during discovery.

67. As just one non-limiting example, set forth below (with claim language in italics) is a description of Defendants' infringement of exemplary claim 16 (including claim 9, upon which claim 16 depends) of the '241 patent in connection with the Wiwynn SV7220G2 Server configured with a Wiwynn NM10GR Mezzanine Card. This description is based on publicly available information. Alacritech reserves the right to modify this description, including, for example, on the basis of information about the '241 Accused Products that it obtains during discovery.

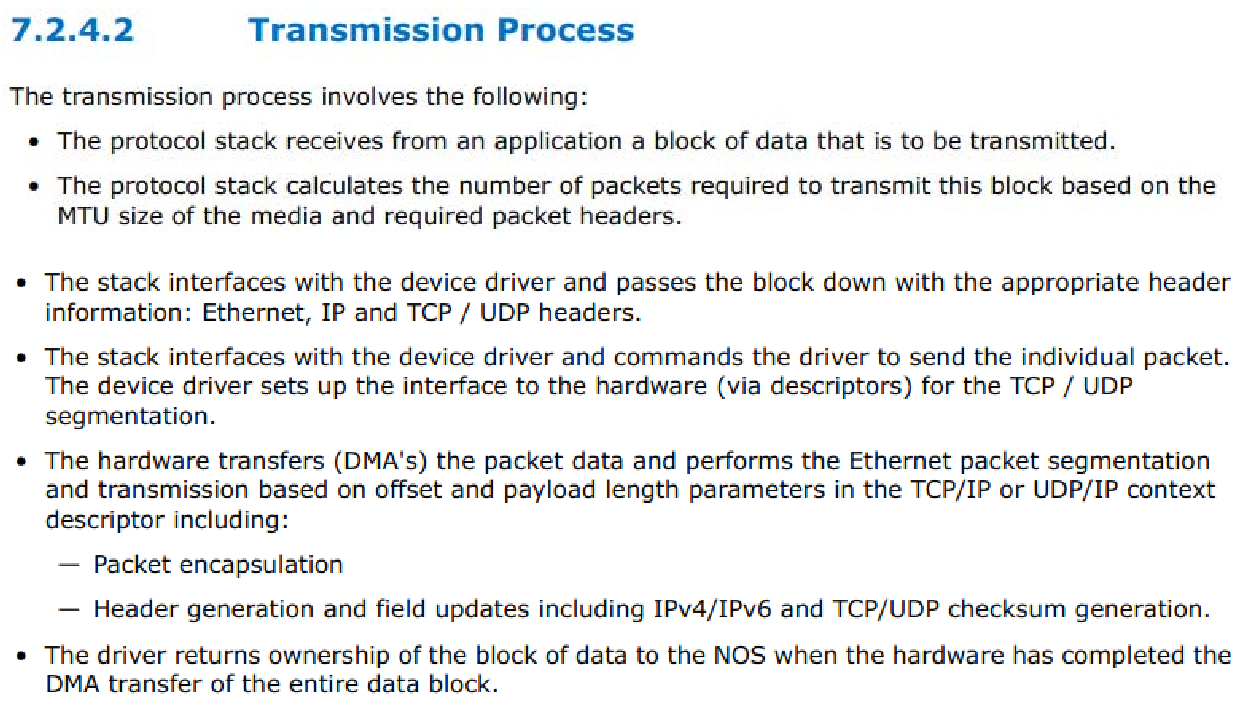
9(a) *A method for communicating information over a network, the method comprising: obtaining data from a source in memory allocated by a first processor; dividing the data into multiple segments;* – Defendants make, use, sell, offer for sale, and/or import servers, network interface devices (e.g., controllers and cards), and other network products that support infringing RSC and LSO functionality. One example of an infringing products is the Wiwynn SV7220G2 Server (the "SV7220G2 Server") configured with a Wiwynn NM10GR Mezzanine Card (the "NM10GR Card") comprising an Intel X550 controller (the "X550 Controller"). *See, e.g.*, "Wiwynn 2016 OCP US Summit Product Brochure," available at https://www.circleb.eu/wp-content/uploads/2016/04/OCP-US-Summit-OPS.pdf ("OCP Product Brochure"); "Wiwynn SV7220G2 Series Datasheet," available at http://www.wiwynn.com/usr_files/20150413140223_Datasheet%20-%20SV7220G2_150413.pdf ("SV7220G2 Datasheet"). A SV7220G2 Server has memory and a first processor (e.g., an Intel Xeon processor). *See, e.g.*, SV7220G2 Datasheet. The "Intel Ethernet Controller X550 Datasheet," Revision 2.1 (May 2016), available at http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/ethernet-x550-datasheet.pdf (the "X550 Datasheet") provides an overview of the X550 Controller and explains how it interacts with a compatible server (such as the SV7220G2 Server), including, for example, describing in Section 7.2 how it communicates data and in Section 7.2.4, in particular, how it provides LSO functionality. As a part of communicating information over a network using LSO functionality, the NM10GR Card of a SV7220G2 Server obtains the data to be sent from a source in server memory allocated by a first processor of the server, and the data is also divided into segments that comply with the Maximum Transmission Unit of the network medium. This is explained, for example, in the Section 7.2.4 introduction and in Section 7.2.4.2 of the X550 Datasheet.



7.2.4 TCP and UDP Segmentation

Hardware TCP segmentation is one of the offloading options supported by the Windows* and Linux* TCP/IP stack. This is often referred to as Large Send offloading or TSO. This feature enables the TCP/IP stack to pass to the network device driver a message to be transmitted that is bigger than the Maximum Transmission Unit (MTU) of the medium. It is then the responsibility of the device driver and hardware to divide the TCP message into MTU size frames that have appropriate layer 2 (Ethernet), 3 (IP), and 4 (TCP) headers. These headers must include sequence number, checksum fields, options and flag values as required. Note that some of these values (such as the checksum values) are unique for each packet of the TCP message, and other fields such as the source IP Address is constant for all packets associated with the TCP message.

***



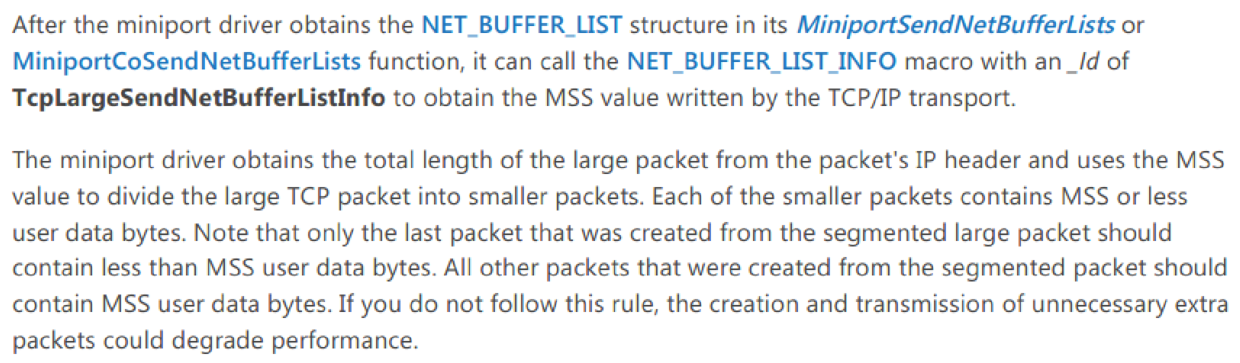
7.2.4.2 Transmission Process

The transmission process involves the following:

- The protocol stack receives from an application a block of data that is to be transmitted.
- The protocol stack calculates the number of packets required to transmit this block based on the MTU size of the media and required packet headers.

- The stack interfaces with the device driver and passes the block down with the appropriate header information: Ethernet, IP and TCP / UDP headers.
- The stack interfaces with the device driver and commands the driver to send the individual packet. The device driver sets up the interface to the hardware (via descriptors) for the TCP / UDP segmentation.
- The hardware transfers (DMA's) the packet data and performs the Ethernet packet segmentation and transmission based on offset and payload length parameters in the TCP/IP or UDP/IP context descriptor including:
  - — Packet encapsulation
  - — Header generation and field updates including IPv4/IPv6 and TCP/UDP checksum generation.
- The driver returns ownership of the block of data to the NOS when the hardware has completed the DMA transfer of the entire data block.

Microsoft also describes this functionality for a network interface device supporting infringing LSO functionality operating with a server (such as the SV7220G2 Server incorporating the NM10GR Card) running the Microsoft Windows Server Operating System, which enables LSO by default. *See, e.g.*, "Offloading the Segmentation of Large TCP Packets," available at https://msdn.microsoft.com/en-us/library/windows/hardware/ff568840.aspx.


After the miniport driver obtains the NET_BUFFER_LIST structure in its *MiniportSendNetBufferLists* or MiniportCoSendNetBufferLists function, it can call the NET_BUFFER_LIST_INFO macro with an _*Id* of **TcpLargeSendNetBufferListInfo** to obtain the MSS value written by the TCP/IP transport.

The miniport driver obtains the total length of the large packet from the packet's IP header and uses the MSS value to divide the large TCP packet into smaller packets. Each of the smaller packets contains MSS or less user data bytes. Note that only the last packet that was created from the segmented large packet should contain less than MSS user data bytes. All other packets that were created from the segmented packet should contain MSS user data bytes. If you do not follow this rule, the creation and transmission of unnecessary extra packets could degrade performance.

9(b) *prepending a packet header to each of the segments by a second processor, thereby forming a packet corresponding to each segment, each packet header containing a media access control layer header, a network layer header and a transport layer header, wherein the network layer header is Internet Protocol (IP), the transport layer header is Transmission Control Protocol (TCP) and the media access control layer header, the network layer header and the transport layer header are prepended at one time as a sequence of bits during the prepending of each packet header; and* – In connection with communicating information over a network using the LSO functionality described above, a second processor of the NM10GR Card of a SV7220G2 Server prepends a packet header to each segment to form a corresponding packet for communication. The packet header contains a media access control layer (MAC) header, a network layer header (specifically, an IP header), and a transport layer header (specifically, a TCP header) that are prepended to each segment at one time as a sequence of bits. The X550 Datasheet describes this operation in the sections shown above and, also, for example, in Sections 7.2.4.4 and 7.2.4.7.

7.2.4.4 Packet Format

A TCP / UDP message can be as large as 256 KB and is generally fragmented across multiple pages in host memory. The X550 partitions the data packet into standard Ethernet frames prior to transmission. The X550 supports calculating the Ethernet, IP, TCP, and UDP headers, including checksum, on a frame-by-frame basis. For tunneled packets (NVGRE, VXLAN), The X550 also supports updating the outer IPv4 header.

Table 7-41 TCP/IP and UDP/IP Packet Format Sent by Host

Pseudo Header				Data
Ethernet	Optional tunnel header	IPv4/IPv6	TCP/UDP	DATA (full TCP/UDP message)

Table 7-42 Packets Format Sent by Device

Pseudo Header (updated)	Data (first MSS)	FCS	...	Pseudo Header (updated)	Data (Next MSS)	FCS	...

***

7.2.4.7 IP/TCP/UDP Header Updating

IP/TCP and IP/UDP header is updated for each outgoing frame based on the header prototype that hardware DMA's from the first descriptor(s). The checksum fields and other header information are later updated on a frame-by-frame basis. The updating process is performed concurrently with the packet data fetch.

The following sections define what fields are modified by hardware during the TCP/UDP segmentation process by the X550.

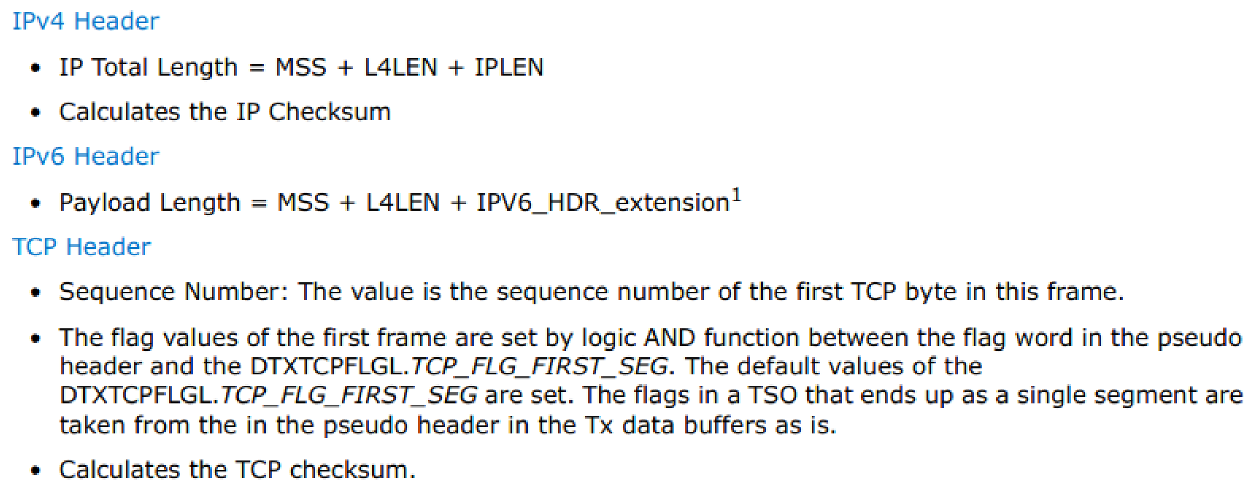
7.2.4.7.1 TCP/IP/UDP Header for the First Frame

The hardware makes the following changes to the headers of the first packet that is derived from each TCP segmentation context.

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follows:

- IP Total Length = OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + MSS
- Calculates the Outer IP Checksum



IPv4 Header

- IP Total Length = MSS + L4LEN + IPLEN
- Calculates the IP Checksum

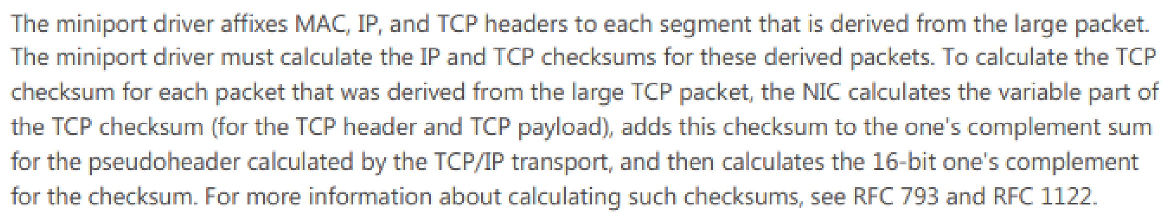
IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension[1]

TCP Header

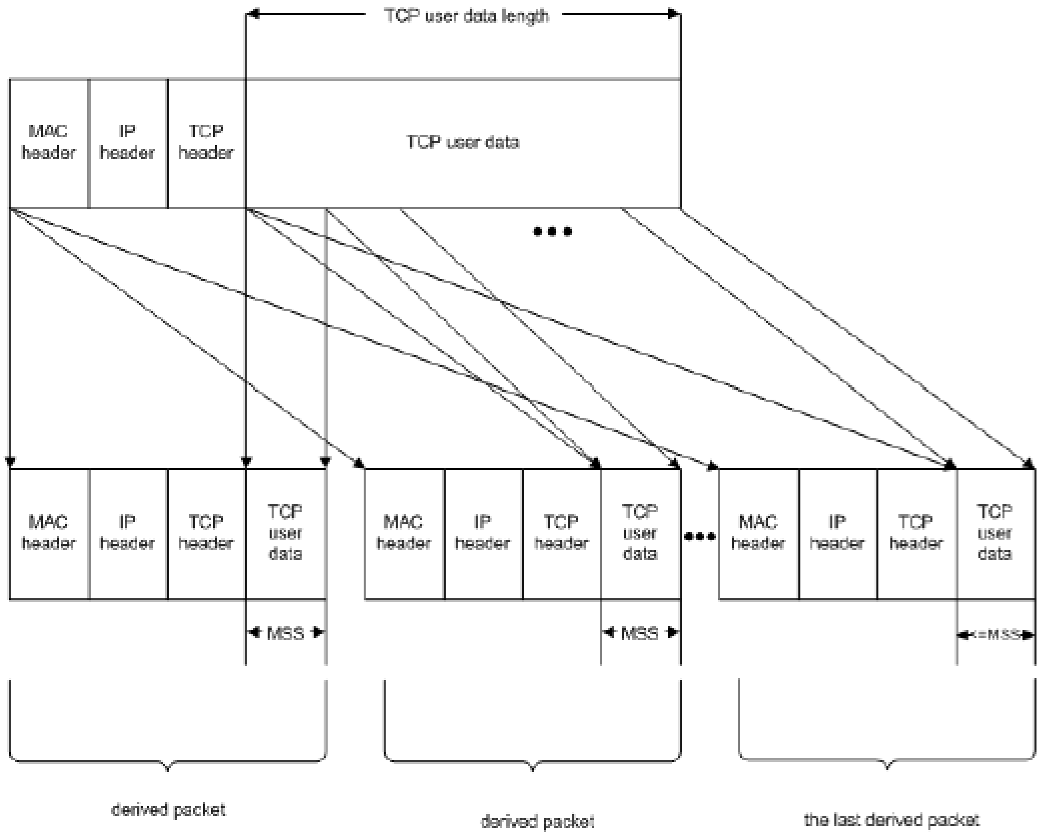
- Sequence Number: The value is the sequence number of the first TCP byte in this frame.
- The flag values of the first frame are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGL.*TCP_FLG_FIRST_SEG*. The default values of the DTXTCPFLGL.*TCP_FLG_FIRST_SEG* are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculates the TCP checksum.

Microsoft also describes and diagrams this operation. *See, e.g.*, "Offloading the Segmentation of Large TCP Packets," available at https://msdn.microsoft.com/en-us/library/windows/hardware/ff568840.aspx.



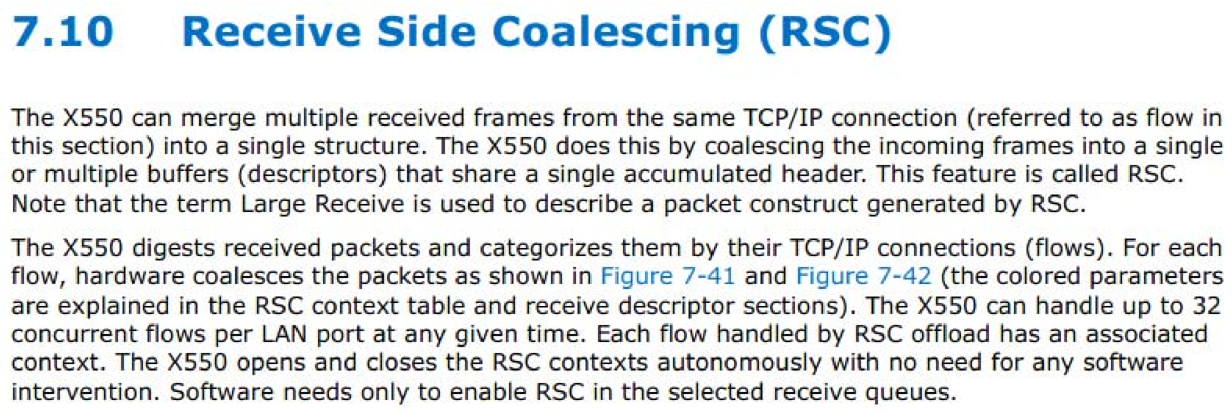
The miniport driver affixes MAC, IP, and TCP headers to each segment that is derived from the large packet. The miniport driver must calculate the IP and TCP checksums for these derived packets. To calculate the TCP checksum for each packet that was derived from the large TCP packet, the NIC calculates the variable part of the TCP checksum (for the TCP header and TCP payload), adds this checksum to the one's complement sum for the pseudoheader calculated by the TCP/IP transport, and then calculates the 16-bit one's complement for the checksum. For more information about calculating such checksums, see RFC 793 and RFC 1122.

The following figure shows the segmentation of a large packet.



9(c) *transmitting the packets to the network.* – As discussed in the materials above, the NM10GR Card of a SV7220G2 Server transmits the packets prepared using LSO functionality to the network.

16(a) *The method of claim 9, further comprising: receiving another packet from the network, the other packet containing a receive header including information corresponding to a network layer and a transport layer; and* – The NM10GR Card of a SV7220G2 Server receives packets from the network that contain a receive header with information corresponding to a network layer (e.g., IP) and a transport layer (e.g., TCP). The X550 Datasheet describes in Section 7.10, for example, how it performs this operation in connection with RSC functionality, and in Section 7.10.1, in particular, the X550 Datasheet includes a table indicating that a packet's receive header includes TCP and IP information.

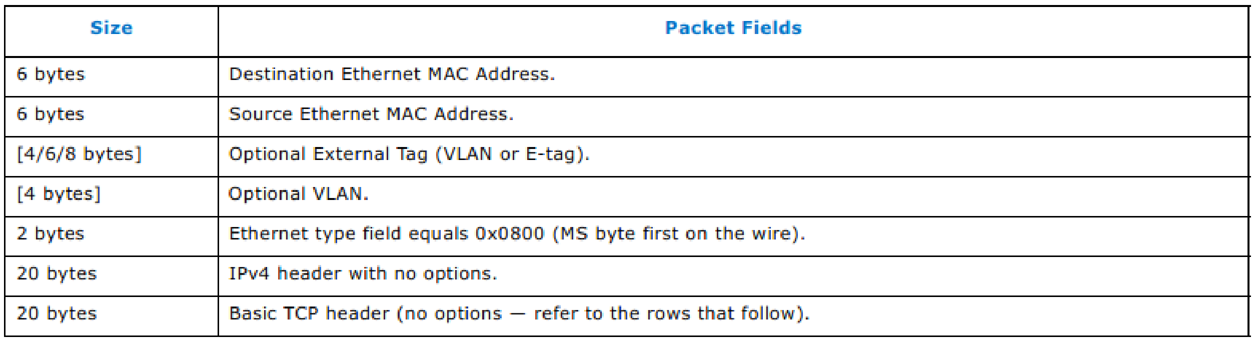


7.10 Receive Side Coalescing (RSC)

The X550 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X550 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

The X550 digests received packets and categorizes them by their TCP/IP connections (flows). For each flow, hardware coalesces the packets as shown in Figure 7-41 and Figure 7-42 (the colored parameters are explained in the RSC context table and receive descriptor sections). The X550 can handle up to 32 concurrent flows per LAN port at any given time. Each flow handled by RSC offload has an associated context. The X550 opens and closes the RSC contexts autonomously with no need for any software intervention. Software needs only to enable RSC in the selected receive queues.

***



Size	Packet Fields
6 bytes	Destination Ethernet MAC Address.
6 bytes	Source Ethernet MAC Address.
[4/6/8 bytes]	Optional External Tag (VLAN or E-tag).
[4 bytes]	Optional VLAN.
2 bytes	Ethernet type field equals 0x0800 (MS byte first on the wire).
20 bytes	IPv4 header with no options.
20 bytes	Basic TCP header (no options — refer to the rows that follow).

16(b) *selecting whether to process the other packet by the first processor or by the second processor.* – In connection with the RSC functionality described above, the NM10GR Card of a SV7220G2 Server selects whether the received packet is processed by the first processor or by the second processor (of the NM10GR Card). As described, for example, in Section 7.10.1 and diagramed in Figure 7-43 of the X550 Datasheet, the NM10GR Card checks whether a number of conditions are met to determine whether the packet is viable for RSC processing (in which case it is processed by the second processor) or must be processed using the legacy (non-coalescing) scheme (in which case it is processed by the first processor).

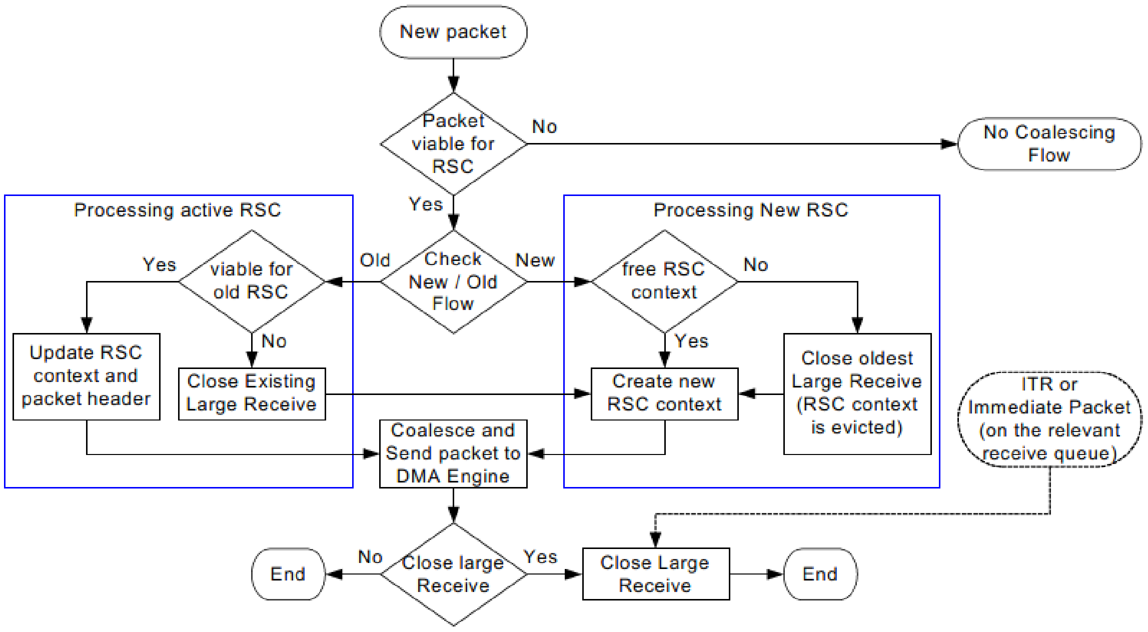


Figure 7-43 RSC Event Flow

7.10.1 Packet Candidacy for RSC

Incoming packets can be good candidates for RSC offload when the following conditions are met. If any of the these conditions are not met, and assuming the queue is configured as required in Section 4.6.7.2 the received packet is processed in the legacy (non-coalescing) scheme.

68. As of no later than the filing and service of this Complaint, Defendants are also indirectly infringing the '241 patent.

69. Defendants have actual knowledge of Alacritech's rights in the '241 patent and details of Defendants' infringement of the '241 patent based on at least the filing and service of this Complaint.

70. Defendants make, use, import, offer for sale, and/or sell the '241 Accused Products (including, for example, the products described above) with knowledge of or willful blindness to the fact that their actions will induce their customers to infringe the '241 patent. Defendants induce others to infringe the '241 patent in violation of 35 U.S.C. § 271 by encouraging and facilitating others to practice the '241 patent's inventions for accelerated network communications with intent that those performing the acts infringe the '241 patent. For

example, Defendants sell and/or provide '241 Accused Products that perform infringing LSO and RSC functionality by default. When Defendants' customers use the '241 Accused Products for their intended purpose, they practice one or more claims of the '241 patent. Defendants instructs their customers how to set up and use '241 Accused Products. In some cases, Defendants even sets up '241 Accused Products for their customers. Once they have been set up, Defendants' customers infringe one or more claims of the '241 patent simply by using the '241 Accused Products. By selling and/or providing their customers '241 Accused Products that infringe by default when used for their intended purpose, and by setting up for their customers and/or instructing their customers how to set up and use those products, Defendants induce those customers to infringe the '241 patent.

71. Defendants also contribute to the infringement of the '241 patent in violation of 35 U.S.C. § 271. Defendants know that infringing components of the '241 Accused Products are especially made or especially adapted for use in the infringement of the '241 patent. The infringing components of these products are not staple articles or commodities of commerce suitable for substantial non-infringing use, and the infringing components of these products are a material part of the invention of the '241 patent. The '241 Accused Products contain infringing components such as specialized hardware and/or software for performing infringing LSO or RSC functionality, which are separable from the '241 Accused Products, material to practicing the '241 patent's inventions for accelerated network communications, and have no substantial non-infringing use. Accordingly, Defendants are also contributing to the direct infringement of the '241 patent by their customers.

72. Defendants are not licensed or otherwise authorized by Alacritech to practice, contributorily practice and/or induce third parties to practice the claims of the '241 patent.

73. By reason of Defendants' infringing activities, Alacritech has suffered, and will continue to suffer, substantial damages in an amount to be proven at trial. Alacritech is entitled to a money judgment in an amount adequate to compensate for Defendants' infringement, but in no event less than a reasonable royalty for the use made of the invention by Defendants, together with interest and costs as fixed by the Court.

74. Defendants' continuing acts of infringement have caused and will continue to cause Alacritech irreparable harm, for which Alacritech has no adequate remedy at law, unless Defendants' continuing acts of infringement are enjoined by the Court. The hardships that an injunction would impose are less than those faced by Alacritech should an injunction not issue. The public interest would be served by issuance of an injunction.

75. Defendants' infringement of the '241 patent is exceptional and entitles Alacritech to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

COUNT IV

INFRINGEMENT OF U.S. PATENT NO. 7,673,072

76. Alacritech re-alleges and incorporates by reference each of the allegations of paragraphs 1-36 set forth above as though fully set forth herein.

77. Alacritech is the current exclusive owner and assignee of all right, title, and interest in and to U.S. Patent No. 7,673,072 (the "'072 patent"), titled "Fast-Path Apparatus for Transmitting Data Corresponding to a TCP Connection," duly and legally issued by the United States Patent and Trademark Office on March 2, 2010, including the right to bring this suit for injunctive relief and damages. A true and correct copy of the '072 patent is attached hereto as Exhibit D.

78. The '072 patent is valid and enforceable.

79. Defendants have directly infringed and are currently directly infringing the '072 patent by making, using, selling, offering for sale, and/or importing into the United States, without authority, products, methods, equipment, and/or services that practice one or more claims of the '072 patent in connection with infringing LSO functionality, including but not limited to the Wiwynn SV100G2, SV300, SV300G2, SV320, SV320G2, SV324G2, SV5270G2-R, SV5270G2-S, SV7110, SV7220-2A, SV7220-2P, SV7220-2S, SV7220G2-N, SV7220G2-P, SV7220G2-S, SV7220G2-V, NM10GR, and NM10GS; any other activities, products and/or services involving the products identified above; and any other activities, products and/or services that practice and/or support similarly infringing LSO functionality (collectively, "the '072 Accused Products"). The '072 Accused Products are non-limiting examples that were identified based on publicly available information, and Alacritech reserves the right to identify additional infringing activities, products and services, including, for example, on the basis of information obtained during discovery.

80. As just one non-limiting example, set forth below (with claim language in italics) is a description of Defendants' infringement of exemplary claim 16 (including claim 15, upon which claim 16 depends) of the '072 patent in connection with the Wiwynn SV7220G2 Server configured with a Wiwynn NM10GR Mezzanine Card. This description is based on publicly available information. Alacritech reserves the right to modify this description, including, for example, on the basis of information about the '072 Accused Products that it obtains during discovery.

15(a) *A method comprising: establishing, at a computer, a Transmission Control Protocol (TCP) connection corresponding to a context that includes status information and Internet Protocol (IP) addresses and TCP ports for the connection; transferring the context to an* *interface device;* - Defendants make, use, sell, offer for sale, and/or import servers, network interface devices (e.g., controllers and cards), and other network products that support infringing LSO functionality. One example of an infringing products is the Wiwynn SV7220G2 Server (the "SV7220G2 Server") configured with a Wiwynn NM10GR Mezzanine Card (the "NM10GR Card") comprising an Intel X550 controller (the "X550 Controller"). *See, e.g.*, "Wiwynn 2016 OCP US Summit Product Brochure," available at https://www.circleb.eu/wp-content/uploads/2016/04/OCP-US-Summit-OPS.pdf ("OCP Product Brochure"); "Wiwynn SV7220G2 Series Datasheet," available at http://www.wiwynn.com/usr_files/20150413140223_Datasheet%20-%20SV7220G2_150413.pdf ("SV7220G2 Datasheet"). In connection with LSO functionality, a SV7220G2 Server establishes a Transmission Control Protocol (TCP) connection that corresponds to a context for the communication including status information, Internet Protocol (IP) addresses, and TCP ports, and transfers the context to an interface device comprising the NM10GR Card with X550 Controller. The "Intel Ethernet Controller X550 Datasheet," Revision 2.1 (May 2016), available at http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/ethernet-x550-datasheet.pdf (the "X550 Datasheet") provides an overview of the X550 Controller and explains how it interacts with the server, including, for example, describing in Section 7.2.4 (excerpts of which appear below) how it segments and sends data according to a TCP connection established by the server's TCP/IP stack using corresponding context received from the server that includes status information, IP addresses, and TCP ports for the connection.

7.2.4 TCP and UDP Segmentation

Hardware TCP segmentation is one of the offloading options supported by the Windows* and Linux* TCP/IP stack. This is often referred to as Large Send offloading or TSO. This feature enables the TCP/IP stack to pass to the network device driver a message to be transmitted that is bigger than the Maximum Transmission Unit (MTU) of the medium. It is then the responsibility of the device driver and hardware to divide the TCP message into MTU size frames that have appropriate layer 2 (Ethernet), 3 (IP), and 4 (TCP) headers. These headers must include sequence number, checksum fields, options and flag values as required. Note that some of these values (such as the checksum values) are unique for each packet of the TCP message, and other fields such as the source IP Address is constant for all packets associated with the TCP message.

***

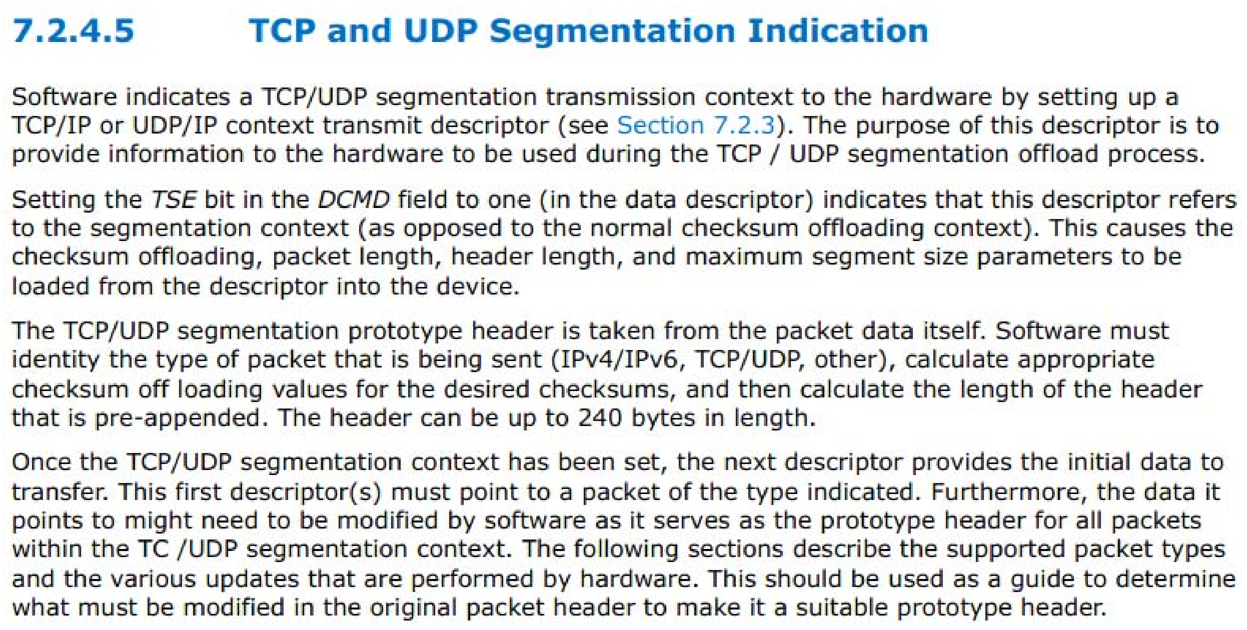
7.2.4.2 Transmission Process

The transmission process involves the following:

- The protocol stack receives from an application a block of data that is to be transmitted.
- The protocol stack calculates the number of packets required to transmit this block based on the MTU size of the media and required packet headers.

- The stack interfaces with the device driver and passes the block down with the appropriate header information: Ethernet, IP and TCP / UDP headers.
- The stack interfaces with the device driver and commands the driver to send the individual packet. The device driver sets up the interface to the hardware (via descriptors) for the TCP / UDP segmentation.
- The hardware transfers (DMA's) the packet data and performs the Ethernet packet segmentation and transmission based on offset and payload length parameters in the TCP/IP or UDP/IP context descriptor including:
  - Packet encapsulation
  - Header generation and field updates including IPv4/IPv6 and TCP/UDP checksum generation.
- The driver returns ownership of the block of data to the NOS when the hardware has completed the DMA transfer of the entire data block.

***



7.2.4.5 TCP and UDP Segmentation Indication

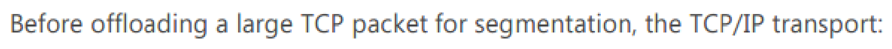
Software indicates a TCP/UDP segmentation transmission context to the hardware by setting up a TCP/IP or UDP/IP context transmit descriptor (see Section 7.2.3). The purpose of this descriptor is to provide information to the hardware to be used during the TCP / UDP segmentation offload process.

Setting the *TSE* bit in the *DCMD* field to one (in the data descriptor) indicates that this descriptor refers to the segmentation context (as opposed to the normal checksum offloading context). This causes the checksum offloading, packet length, header length, and maximum segment size parameters to be loaded from the descriptor into the device.

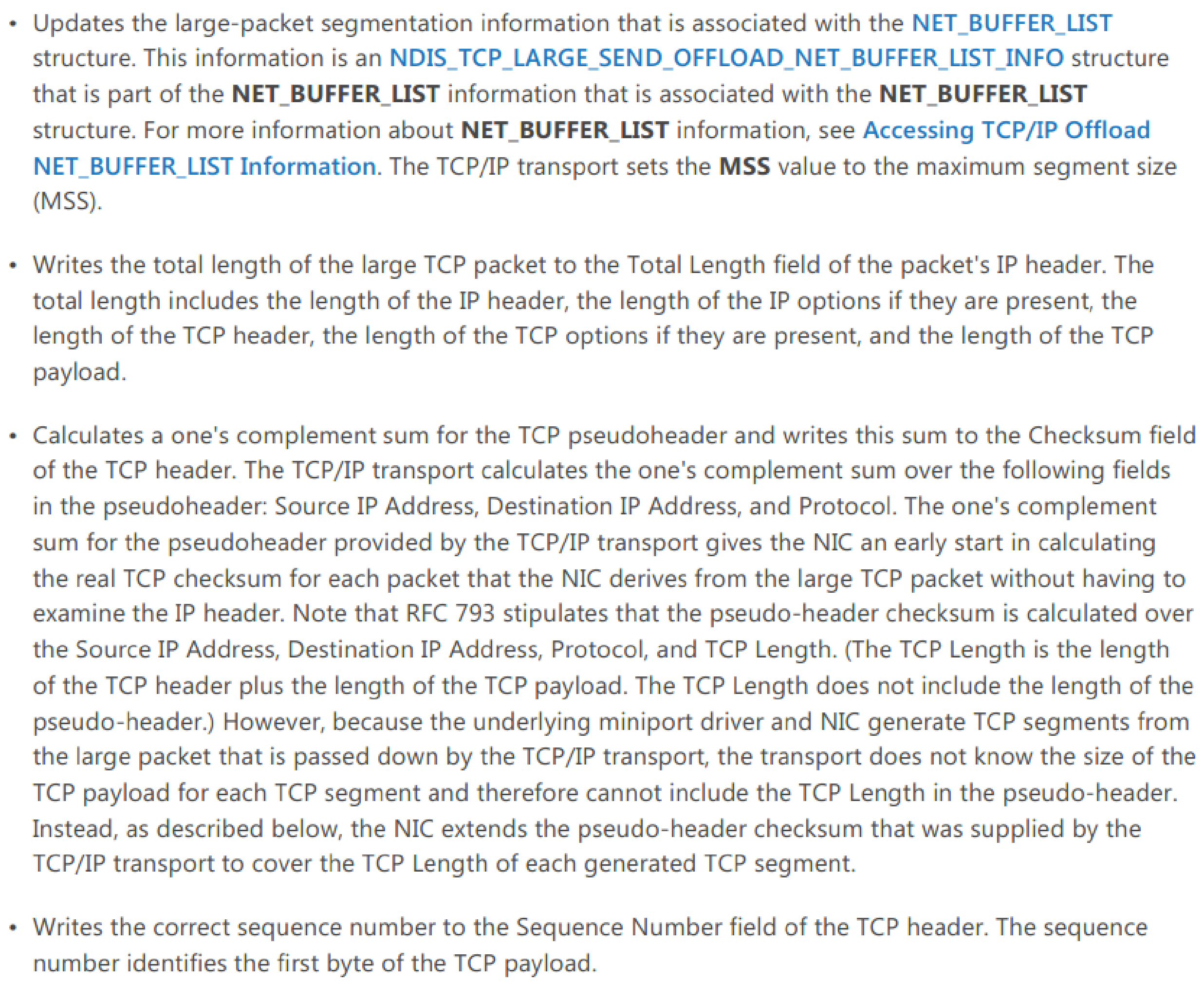
The TCP/UDP segmentation prototype header is taken from the packet data itself. Software must identity the type of packet that is being sent (IPv4/IPv6, TCP/UDP, other), calculate appropriate checksum off loading values for the desired checksums, and then calculate the length of the header that is pre-appended. The header can be up to 240 bytes in length.

Once the TCP/UDP segmentation context has been set, the next descriptor provides the initial data to transfer. This first descriptor(s) must point to a packet of the type indicated. Furthermore, the data it points to might need to be modified by software as it serves as the prototype header for all packets within the TC /UDP segmentation context. The following sections describe the supported packet types and the various updates that are performed by hardware. This should be used as a guide to determine what must be modified in the original packet header to make it a suitable prototype header.

Microsoft also describes this interaction between a network interface device supporting infringing LSO functionality and the TCP/IP stack on a server (such as the SV7220G2 Server incorporating the NM10GR Card) running the Microsoft Windows Server Operating System, which enables LSO by default, including the processing described below that is performed by the TCP/IP stack in connection with LSO. *See*, *e.g.*, "Offloading the Segmentation of Large TCP Packets," available at https://msdn.microsoft.com/en-us/library/windows/hardware/ff568840.aspx.



Before offloading a large TCP packet for segmentation, the TCP/IP transport:



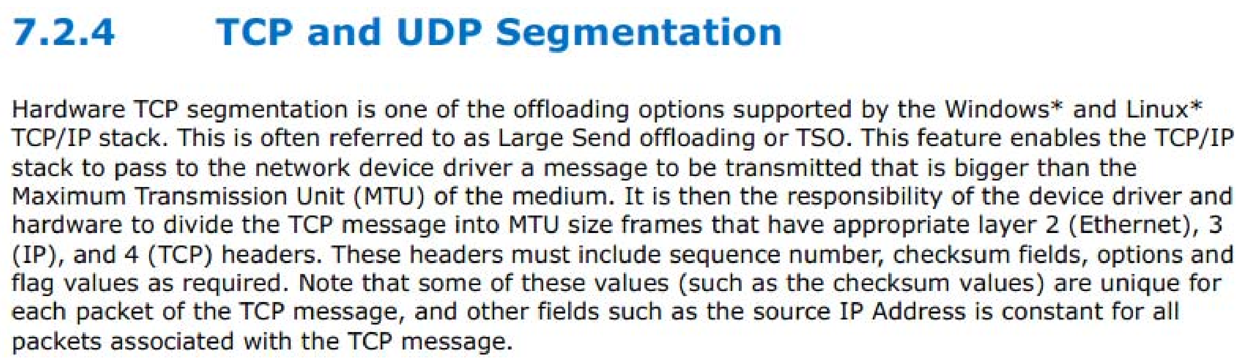
- Updates the large-packet segmentation information that is associated with the NET_BUFFER_LIST structure. This information is an NDIS_TCP_LARGE_SEND_OFFLOAD_NET_BUFFER_LIST_INFO structure that is part of the **NET_BUFFER_LIST** information that is associated with the **NET_BUFFER_LIST** structure. For more information about **NET_BUFFER_LIST** information, see Accessing TCP/IP Offload NET_BUFFER_LIST Information. The TCP/IP transport sets the **MSS** value to the maximum segment size (MSS).

- Writes the total length of the large TCP packet to the Total Length field of the packet's IP header. The total length includes the length of the IP header, the length of the IP options if they are present, the length of the TCP header, the length of the TCP options if they are present, and the length of the TCP payload.

- Calculates a one's complement sum for the TCP pseudoheader and writes this sum to the Checksum field of the TCP header. The TCP/IP transport calculates the one's complement sum over the following fields in the pseudoheader: Source IP Address, Destination IP Address, and Protocol. The one's complement sum for the pseudoheader provided by the TCP/IP transport gives the NIC an early start in calculating the real TCP checksum for each packet that the NIC derives from the large TCP packet without having to examine the IP header. Note that RFC 793 stipulates that the pseudo-header checksum is calculated over the Source IP Address, Destination IP Address, Protocol, and TCP Length. (The TCP Length is the length of the TCP header plus the length of the TCP payload. The TCP Length does not include the length of the pseudo-header.) However, because the underlying miniport driver and NIC generate TCP segments from the large packet that is passed down by the TCP/IP transport, the transport does not know the size of the TCP payload for each TCP segment and therefore cannot include the TCP Length in the pseudo-header. Instead, as described below, the NIC extends the pseudo-header checksum that was supplied by the TCP/IP transport to cover the TCP Length of each generated TCP segment.

- Writes the correct sequence number to the Sequence Number field of the TCP header. The sequence number identifies the first byte of the TCP payload.

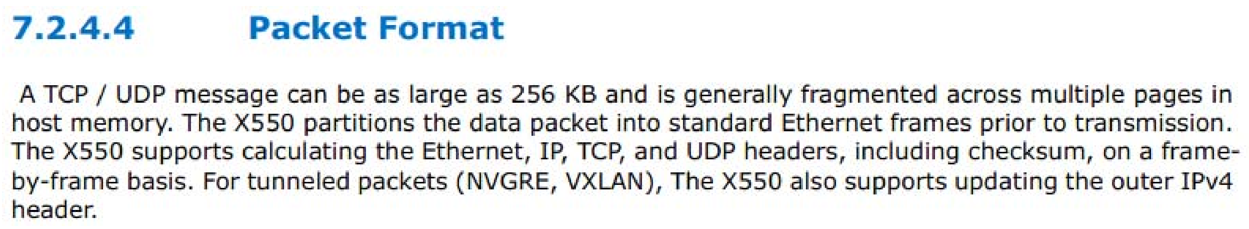
15(b) *transferring data from the network host to the interface device; dividing, by the interface device, the data into segments* – In connection with the LSO functionality discussed above, the SV7220G2 Server incorporating the NM10GR Card transfers data from host memory to the interface device, which divides the data into smaller segments. The X550 Datasheet describes this operation in the sections shown above and, for example, in the Section 7.2.4 introduction and in Section 7.2.4.4.



7.2.4 TCP and UDP Segmentation

Hardware TCP segmentation is one of the offloading options supported by the Windows* and Linux* TCP/IP stack. This is often referred to as Large Send offloading or TSO. This feature enables the TCP/IP stack to pass to the network device driver a message to be transmitted that is bigger than the Maximum Transmission Unit (MTU) of the medium. It is then the responsibility of the device driver and hardware to divide the TCP message into MTU size frames that have appropriate layer 2 (Ethernet), 3 (IP), and 4 (TCP) headers. These headers must include sequence number, checksum fields, options and flag values as required. Note that some of these values (such as the checksum values) are unique for each packet of the TCP message, and other fields such as the source IP Address is constant for all packets associated with the TCP message.

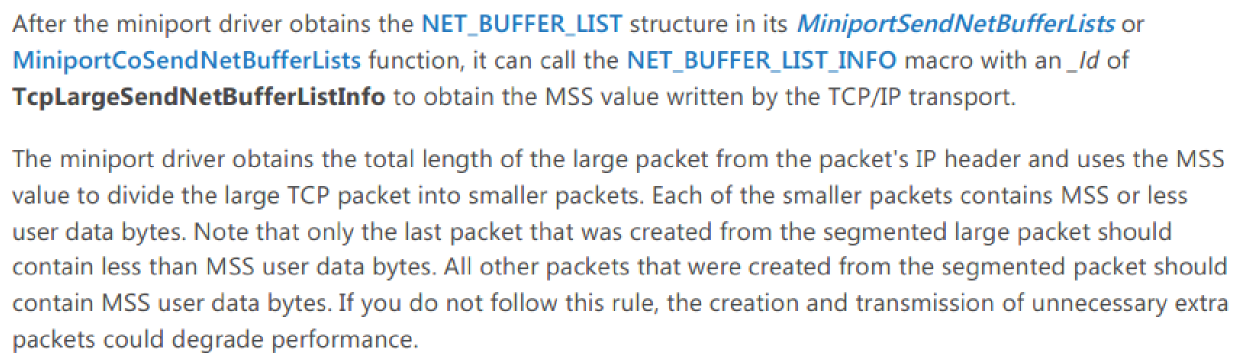
***



7.2.4.4 Packet Format

A TCP / UDP message can be as large as 256 KB and is generally fragmented across multiple pages in host memory. The X550 partitions the data packet into standard Ethernet frames prior to transmission. The X550 supports calculating the Ethernet, IP, TCP, and UDP headers, including checksum, on a frame-by-frame basis. For tunneled packets (NVGRE, VXLAN), The X550 also supports updating the outer IPv4 header.

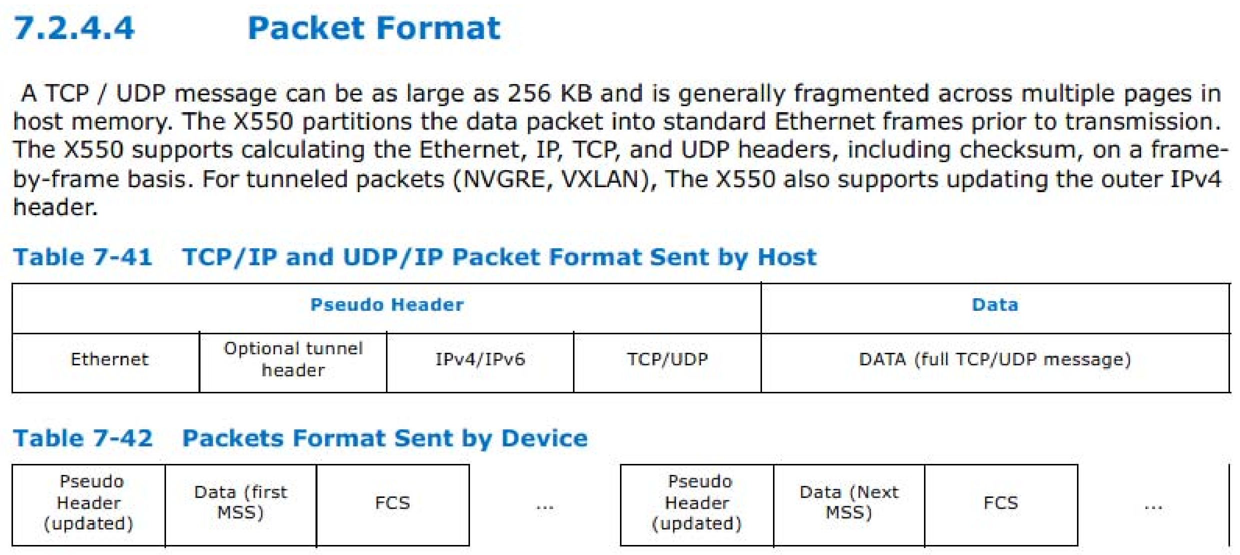
Microsoft also describes this operation. *See*, *e.g.*, "Offloading the Segmentation of Large TCP Packets," available at https://msdn.microsoft.com/en-us/library/windows/hardware/ff568840.aspx.



After the miniport driver obtains the NET_BUFFER_LIST structure in its *MiniportSendNetBufferLists* or MiniportCoSendNetBufferLists function, it can call the NET_BUFFER_LIST_INFO macro with an *_Id* of **TcpLargeSendNetBufferListInfo** to obtain the MSS value written by the TCP/IP transport.

The miniport driver obtains the total length of the large packet from the packet's IP header and uses the MSS value to divide the large TCP packet into smaller packets. Each of the smaller packets contains MSS or less user data bytes. Note that only the last packet that was created from the segmented large packet should contain less than MSS user data bytes. All other packets that were created from the segmented packet should contain MSS user data bytes. If you do not follow this rule, the creation and transmission of unnecessary extra packets could degrade performance.

15(c) *creating headers for the segments, by the interface device, from a template header that includes the IP addresses and TCP ports; and prepending the headers to the segments to form transmit packets.* – In connection with the LSO functionality discussed above, the interface device creates headers for the segments from a template header (e.g., a Pseudo Header) that includes IP addresses and TCP ports, and prepends the headers to the segments to form transmit packets. This is described in the sections shows above and, also, for example, in Sections 7.2.4.4 and 7.2.4.7 of the X550 Datasheet.



7.2.4.4 Packet Format

A TCP / UDP message can be as large as 256 KB and is generally fragmented across multiple pages in host memory. The X550 partitions the data packet into standard Ethernet frames prior to transmission. The X550 supports calculating the Ethernet, IP, TCP, and UDP headers, including checksum, on a frame-by-frame basis. For tunneled packets (NVGRE, VXLAN), The X550 also supports updating the outer IPv4 header.

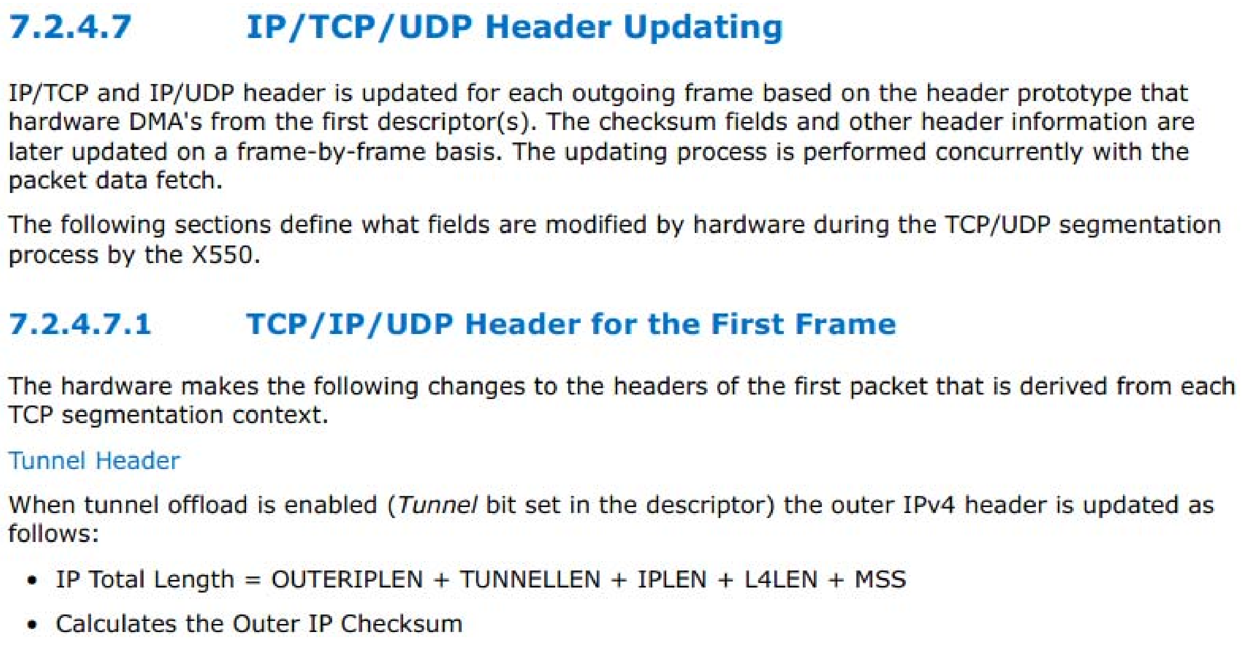
Table 7-41 TCP/IP and UDP/IP Packet Format Sent by Host

Pseudo Header				Data
Ethernet	Optional tunnel header	IPv4/IPv6	TCP/UDP	DATA (full TCP/UDP message)

Table 7-42 Packets Format Sent by Device

Pseudo Header (updated)	Data (first MSS)	FCS	...	Pseudo Header (updated)	Data (Next MSS)	FCS	...

***



7.2.4.7 IP/TCP/UDP Header Updating

IP/TCP and IP/UDP header is updated for each outgoing frame based on the header prototype that hardware DMA's from the first descriptor(s). The checksum fields and other header information are later updated on a frame-by-frame basis. The updating process is performed concurrently with the packet data fetch.

The following sections define what fields are modified by hardware during the TCP/UDP segmentation process by the X550.

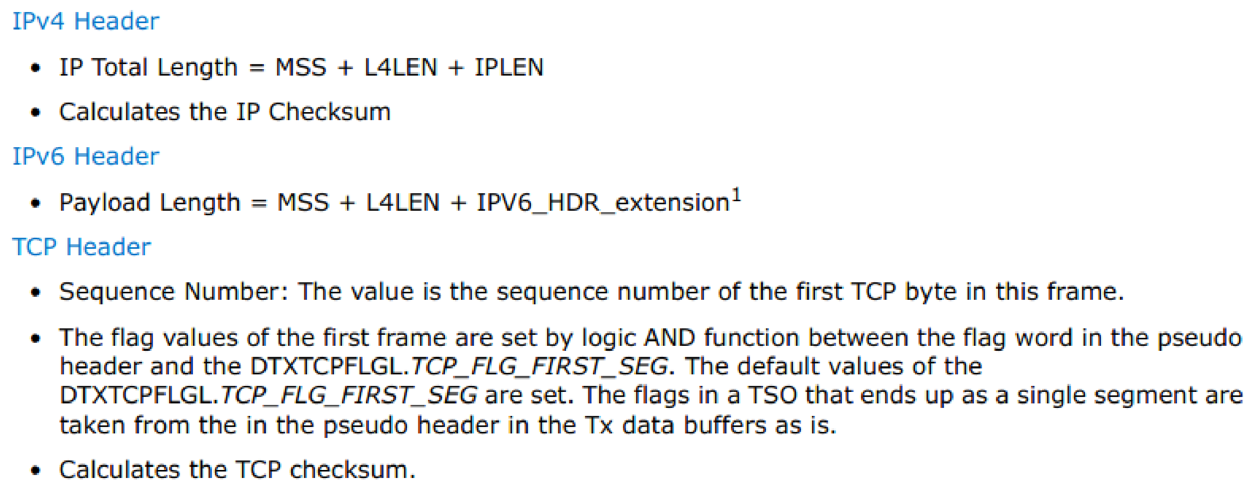
7.2.4.7.1 TCP/IP/UDP Header for the First Frame

The hardware makes the following changes to the headers of the first packet that is derived from each TCP segmentation context.

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follows:

- IP Total Length = OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + MSS
- Calculates the Outer IP Checksum



IPv4 Header

- IP Total Length = MSS + L4LEN + IPLEN
- Calculates the IP Checksum

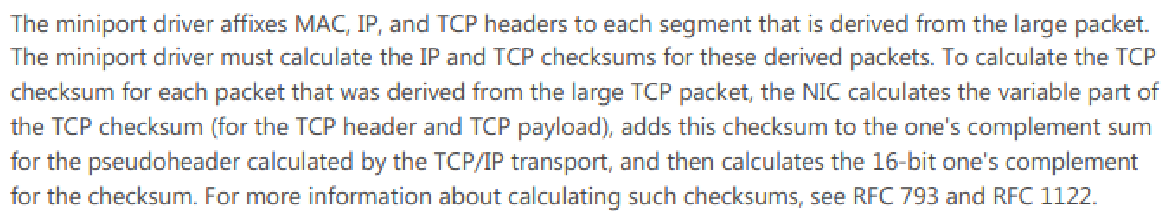
IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension[1]

TCP Header

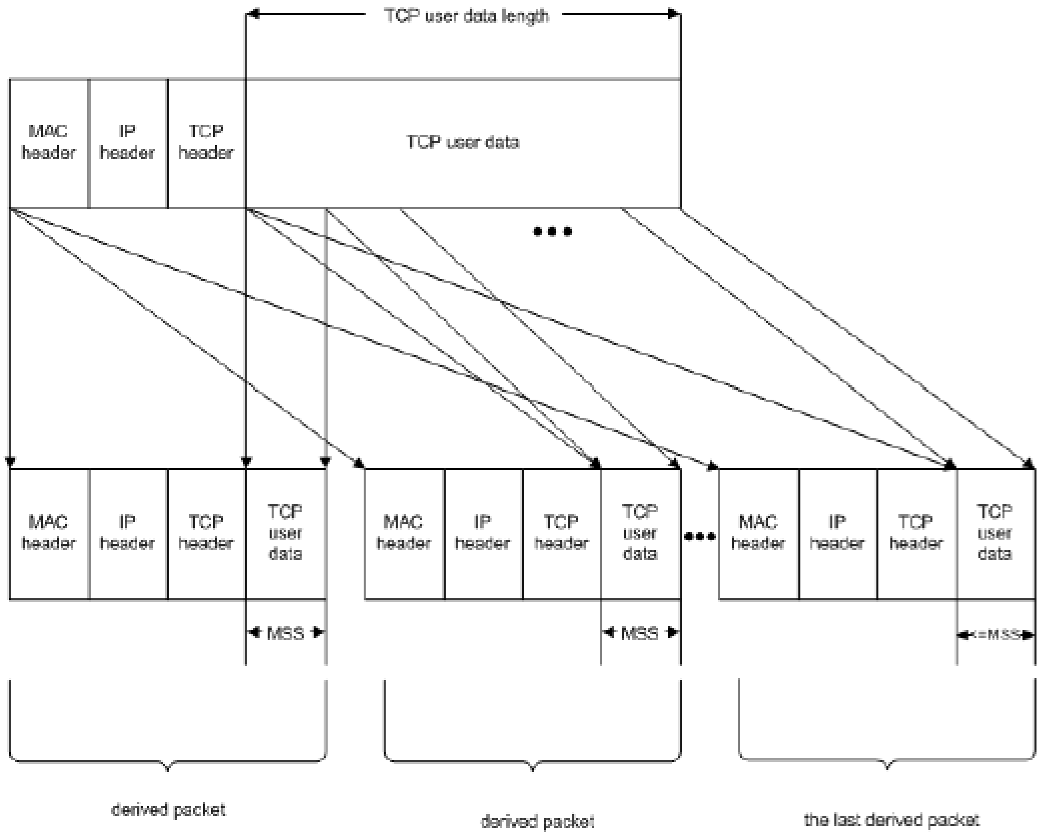
- Sequence Number: The value is the sequence number of the first TCP byte in this frame.
- The flag values of the first frame are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGL.*TCP_FLG_FIRST_SEG*. The default values of the DTXTCPFLGL.*TCP_FLG_FIRST_SEG* are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculates the TCP checksum.

Microsoft also describes and diagrams this operation. *See, e.g.*, "Offloading the Segmentation of Large TCP Packets," available at https://msdn.microsoft.com/en-us/library/windows/hardware/ff568840.aspx.



The miniport driver affixes MAC, IP, and TCP headers to each segment that is derived from the large packet. The miniport driver must calculate the IP and TCP checksums for these derived packets. To calculate the TCP checksum for each packet that was derived from the large TCP packet, the NIC calculates the variable part of the TCP checksum (for the TCP header and TCP payload), adds this checksum to the one's complement sum for the pseudoheader calculated by the TCP/IP transport, and then calculates the 16-bit one's complement for the checksum. For more information about calculating such checksums, see RFC 793 and RFC 1122.

The following figure shows the segmentation of a large packet.

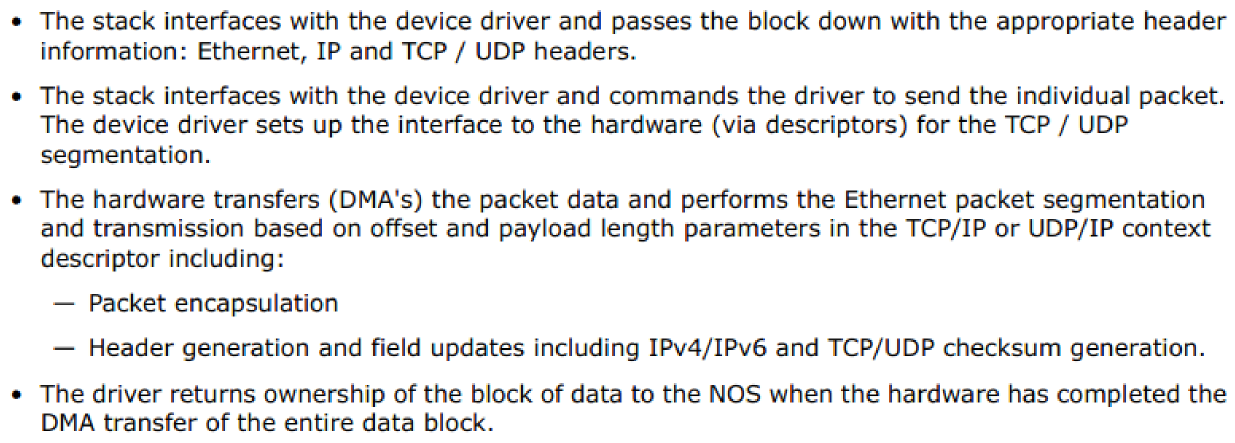


16 *The method of claim 15, wherein transferring the context to the interface device occurs prior to transferring the data to the interface device* – As discussed in the materials above (such as in Sections 7.2.4.2 and 7.2.4.5 of the X550 Datasheet, shown below), the context is transferred to the interface device before the data.

7.2.4.2 Transmission Process

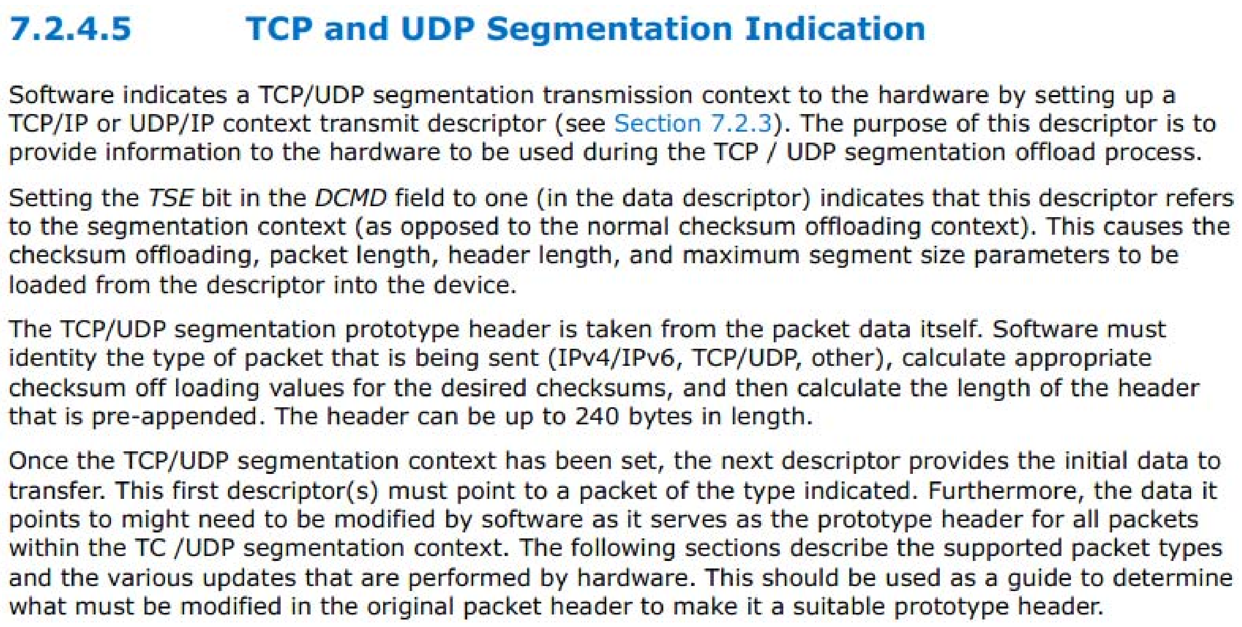
The transmission process involves the following:

- The protocol stack receives from an application a block of data that is to be transmitted.
- The protocol stack calculates the number of packets required to transmit this block based on the MTU size of the media and required packet headers.



- The stack interfaces with the device driver and passes the block down with the appropriate header information: Ethernet, IP and TCP / UDP headers.
- The stack interfaces with the device driver and commands the driver to send the individual packet. The device driver sets up the interface to the hardware (via descriptors) for the TCP / UDP segmentation.
- The hardware transfers (DMA's) the packet data and performs the Ethernet packet segmentation and transmission based on offset and payload length parameters in the TCP/IP or UDP/IP context descriptor including:
  - — Packet encapsulation
  - — Header generation and field updates including IPv4/IPv6 and TCP/UDP checksum generation.
- The driver returns ownership of the block of data to the NOS when the hardware has completed the DMA transfer of the entire data block.

***



7.2.4.5 TCP and UDP Segmentation Indication

Software indicates a TCP/UDP segmentation transmission context to the hardware by setting up a TCP/IP or UDP/IP context transmit descriptor (see Section 7.2.3). The purpose of this descriptor is to provide information to the hardware to be used during the TCP / UDP segmentation offload process.

Setting the *TSE* bit in the *DCMD* field to one (in the data descriptor) indicates that this descriptor refers to the segmentation context (as opposed to the normal checksum offloading context). This causes the checksum offloading, packet length, header length, and maximum segment size parameters to be loaded from the descriptor into the device.

The TCP/UDP segmentation prototype header is taken from the packet data itself. Software must identity the type of packet that is being sent (IPv4/IPv6, TCP/UDP, other), calculate appropriate checksum off loading values for the desired checksums, and then calculate the length of the header that is pre-appended. The header can be up to 240 bytes in length.

Once the TCP/UDP segmentation context has been set, the next descriptor provides the initial data to transfer. This first descriptor(s) must point to a packet of the type indicated. Furthermore, the data it points to might need to be modified by software as it serves as the prototype header for all packets within the TC /UDP segmentation context. The following sections describe the supported packet types and the various updates that are performed by hardware. This should be used as a guide to determine what must be modified in the original packet header to make it a suitable prototype header.

81. As of no later than the filing and service of this Complaint, Defendants are also indirectly infringing the '072 patent.

82. Defendants have actual knowledge of Alacritech's rights in the '072 patent and details of Defendants' infringement of the '072 patent based on at least the filing and service of this Complaint.

83. Defendants make, use, import, offer for sale, and/or sell the '072 Accused Products (including, for example, the products described above) with knowledge of or willful

blindness to the fact that their actions will induce their customers to infringe the '072 patent. Defendants induce others to infringe the '072 patent in violation of 35 U.S.C. § 271 by encouraging and facilitating others to practice the '072 patent's inventions for accelerated network communications with intent that those performing the acts infringe the '072 patent. For example, Defendants sell and/or provide '072 Accused Products that perform infringing LSO functionality by default. When Defendants' customers use the '072 Accused Products for their intended purpose, they practice one or more claims of the '072 patent. Defendants instructs their customers how to set up and use '072 Accused Products. In some cases, Defendants even sets up '072 Accused Products for their customers. Once they have been set up, Defendants' customers infringe one or more claims of the '072 patent simply by using the '072 Accused Products. By selling and/or providing their customers '072 Accused Products that infringe by default when used for their intended purpose, and by setting up for their customers and/or instructing their customers how to set up and use those products, Defendants induce those customers to infringe the '072 patent.

84. Defendants also contribute to the infringement of the '072 patent in violation of 35 U.S.C. § 271. Defendants know that infringing components of the '072 Accused Products are especially made or especially adapted for use in the infringement of the '072 patent. The infringing components of these products are not staple articles or commodities of commerce suitable for substantial non-infringing use, and the infringing components of these products are a material part of the invention of the '072 patent. The '072 Accused Products contain infringing components such as specialized hardware and/or software for performing infringing LSO functionality, which are separable from the '072 Accused Products, material to practicing the '072 patent's inventions for accelerated network communications, and have no substantial non-

infringing use. Accordingly, Defendants are also contributing to the direct infringement of the '072 patent by their customers.

85. Defendants are not licensed or otherwise authorized by Alacritech to practice, contributorily practice and/or induce third parties to practice the claims of the '072 patent.

86. By reason of Defendants' infringing activities, Alacritech has suffered, and will continue to suffer, substantial damages in an amount to be proven at trial. Alacritech is entitled to a money judgment in an amount adequate to compensate for Defendants' infringement, but in no event less than a reasonable royalty for the use made of the invention by Defendants, together with interest and costs as fixed by the Court.

87. Defendants' continuing acts of infringement have caused and will continue to cause Alacritech irreparable harm, for which Alacritech has no adequate remedy at law, unless Defendants' continuing acts of infringement are enjoined by the Court. The hardships that an injunction would impose are less than those faced by Alacritech should an injunction not issue. The public interest would be served by issuance of an injunction.

88. Defendants' infringement of the '072 patent is exceptional and entitles Alacritech to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

COUNT V

INFRINGEMENT OF U.S. PATENT NO. 8,131,880

89. Alacritech re-alleges and incorporates by reference each of the allegations of paragraphs 1-36 set forth above as though fully set forth herein.

90. Alacritech is the current exclusive owner and assignee of all right, title, and interest in and to U.S. Patent No. 8,131,880 (the "'880 patent"), titled "Intelligent Network Interface Device and System for Accelerated Communication," duly and legally issued by the United States Patent and Trademark Office on March 6, 2012, including the right to bring this

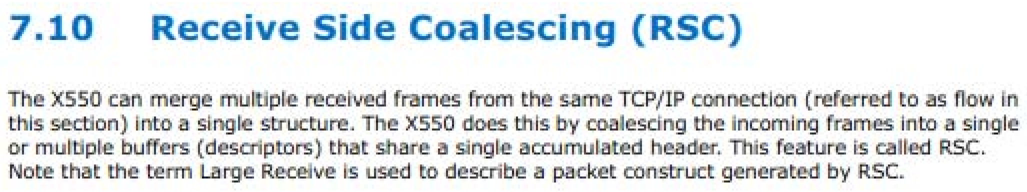
suit for injunctive relief and damages. A true and correct copy of the '880 patent is attached hereto as Exhibit E.

91. The '880 patent is valid and enforceable.

92. Defendants have directly infringed and are currently directly infringing the '880 patent by making, using, selling, offering for sale, and/or importing into the United States, without authority, products, methods, equipment, and/or services that practice one or more claims of the '880 patent in connection with infringing RSC functionality, including but not limited to the Wiwynn SV100G2, SV300, SV300G2, SV320, SV320G2, SV324G2, SV5270G2-R, SV5270G2-S, SV7110, SV7220G2-N, SV7220G2-P, SV7220G2-S, SV7220G2-V, NM10GR, and NM10GS; any other activities, products and/or services involving the products identified above; and any other activities, products and/or services that practice and/or support similarly infringing RSC functionality (collectively, "the '880 Accused Products"). The '880 Accused Products are non-limiting examples that were identified based on publicly available information, and Alacritech reserves the right to identify additional infringing activities, products and services, including, for example, on the basis of information obtained during discovery.

93. As just one non-limiting example, set forth below (with claim language in italics) is a description of Defendants' infringement of exemplary claim 1 of the '880 patent in connection with the Wiwynn SV7220G2 Server configured with a Wiwynn NM10GR Mezzanine Card. This description is based on publicly available information. Alacritech reserves the right to modify this description, including, for example, on the basis of information about the '880 Accused Products that it obtains during discovery.

1(a) *A method of transferring a packet to a host computer system, wherein the packet is received at a communication device from a network, comprising:* – Defendants make, use, sell, offer for sale, and/or import servers, network interface devices (e.g., controllers and cards), and other network products that support infringing RSC functionality. One example of an infringing products is the Wiwynn SV7220G2 Server (the "SV7220G2 Server") configured with a Wiwynn NM10GR Mezzanine Card (the "NM10GR Card") comprising an Intel X550 controller (the "X550 Controller"). *See, e.g.*, "Wiwynn 2016 OCP US Summit Product Brochure," available at https://www.circleb.eu/wp-content/uploads/2016/04/OCP-US-Summit-OPS.pdf ("OCP Product Brochure"); "Wiwynn SV7220G2 Series Datasheet," available at http://www.wiwynn.com/usr_files/20150413140223_Datasheet%20-%20SV7220G2_150413.pdf ("SV7220G2 Datasheet"). To the extent the preamble is limiting, a SV7220G2 Server comprises a host computer system that receives packets from a network at a communication device comprising the NM10GR Card with X550 Controller. *See, e.g.*, OCP Product Brochure; SV7220G2 Datasheet. The "Intel Ethernet Controller X550 Datasheet," Revision 2.1 (May 2016), available at http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/ethernet-x550-datasheet.pdf (the "X550 Datasheet") provides an overview of the X550 Controller and explains how it interacts with the server, including, for example, describing in Section 7.10, Figure 7-41 (an elsewhere) how it digests and coalesces packets it receives from a network in connection with transferring the packet to the host computer system using RSC functionality.



7.10 Receive Side Coalescing (RSC)

The X550 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X550 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

***

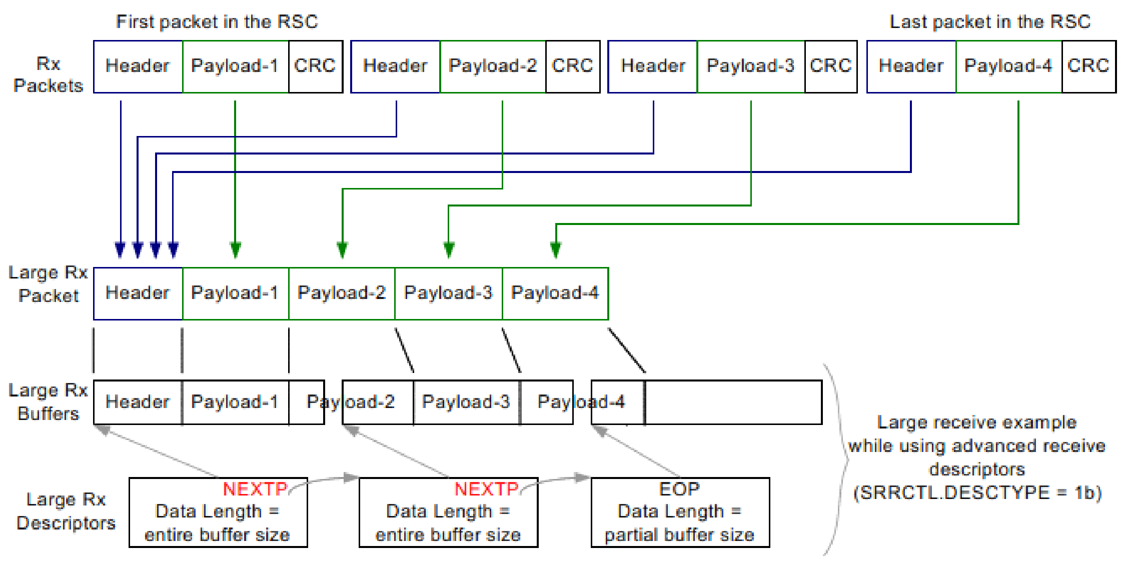
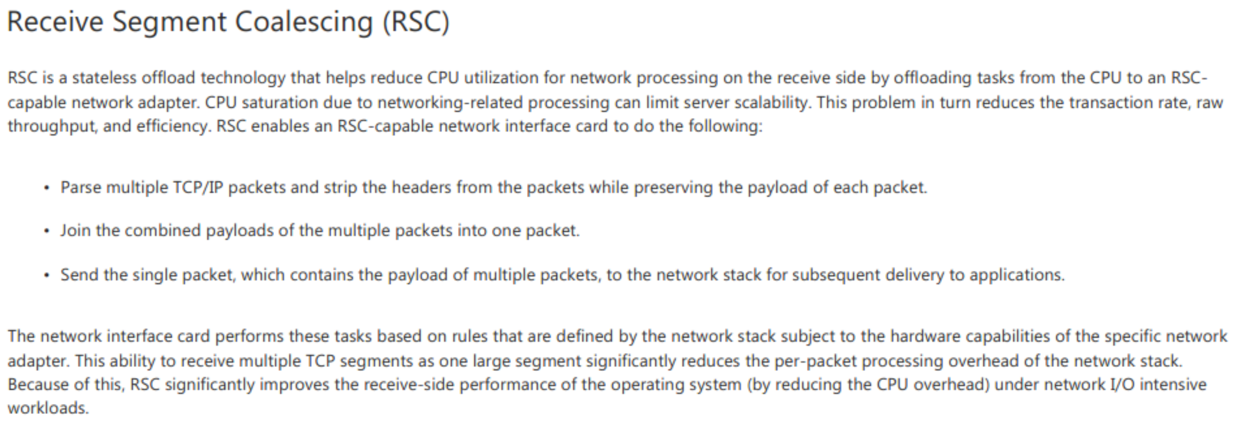


Figure 7-41 RSC Functionality (No Header Split)

Microsoft also describes how a network interface device supporting infringing RSC functionality carries out these interactions with a server (such as the SV7220G2 Server incorporating the NM10GR Card) running the Microsoft Windows Server Operating System, which enables RSC by default. *See*, *e.g.*, "Receive Segment Coalescing," available at https://technet.microsoft.com/en-us/library/hh997024.aspx.



Receive Segment Coalescing (RSC)

RSC is a stateless offload technology that helps reduce CPU utilization for network processing on the receive side by offloading tasks from the CPU to an RSC-capable network adapter. CPU saturation due to networking-related processing can limit server scalability. This problem in turn reduces the transaction rate, raw throughput, and efficiency. RSC enables an RSC-capable network interface card to do the following:

- Parse multiple TCP/IP packets and strip the headers from the packets while preserving the payload of each packet.
- Join the combined payloads of the multiple packets into one packet.
- Send the single packet, which contains the payload of multiple packets, to the network stack for subsequent delivery to applications.

The network interface card performs these tasks based on rules that are defined by the network stack subject to the hardware capabilities of the specific network adapter. This ability to receive multiple TCP segments as one large segment significantly reduces the per-packet processing overhead of the network stack. Because of this, RSC significantly improves the receive-side performance of the operating system (by reducing the CPU overhead) under network I/O intensive workloads.

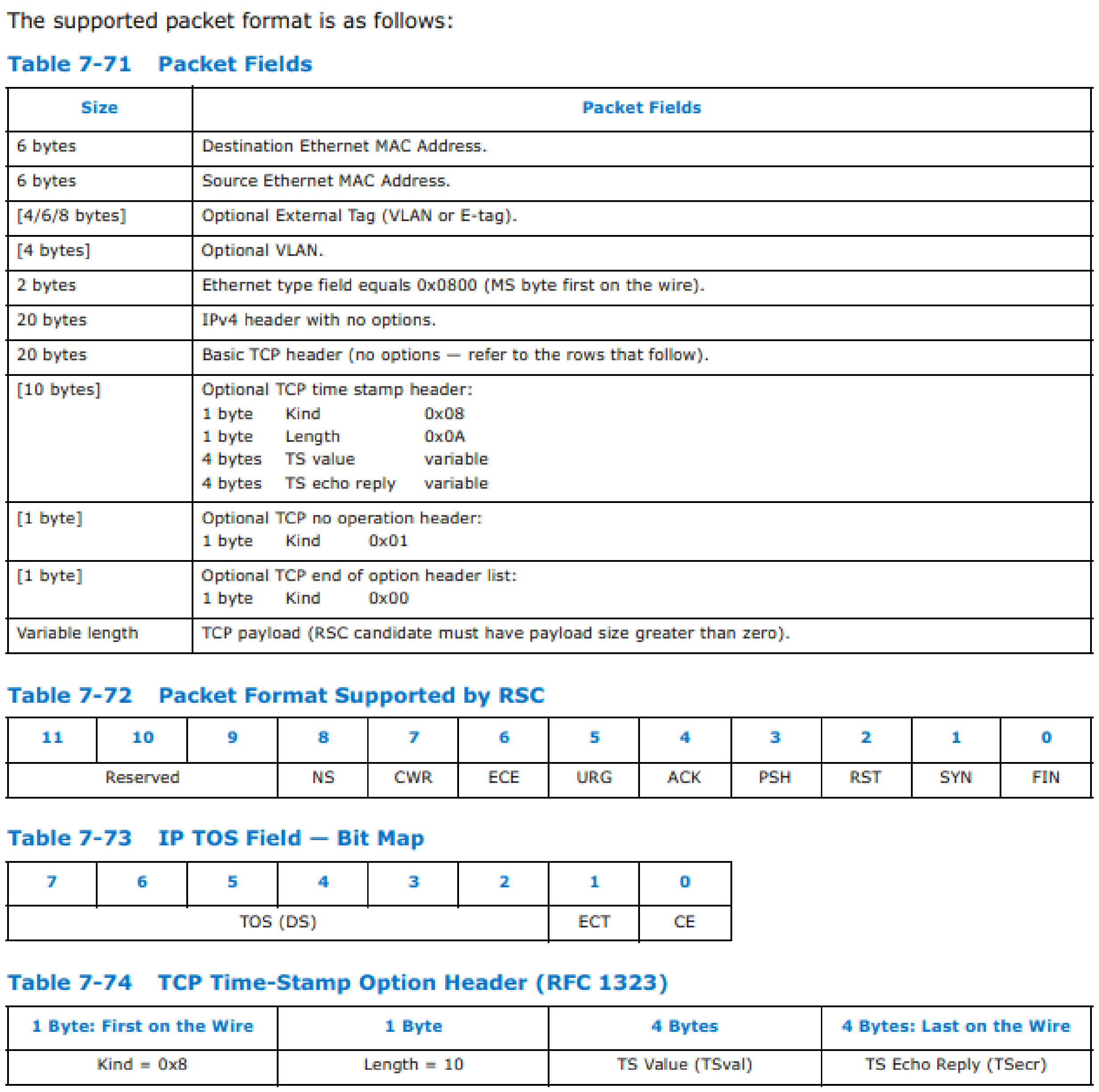
1(b) *parsing a header portion of a first packet received at a network interface for the host computer system to determine if said first packet conforms to a TCP protocol;* – In connection with the RSC functionality discussed above, the SV7220G2 Server incorporating the NM10GR

Card receives a first packet at a network interface for the host computer system, and the NM10GR Card parses a header portion of the packet. The NM10GR Card digests an incoming packet received from the network, and parses the packet header to determine if the packet conforms to a TCP protocol, including inspecting the packet type and the conformance of selected TCP protocol features. This is described, for example, in Section 7.10.1 of the X550 Datasheet.

7.10.1 Packet Candidacy for RSC

Incoming packets can be good candidates for RSC offload when the following conditions are met. If any of the these conditions are not met, and assuming the queue is configured as required in Section 4.6.7.2 the received packet is processed in the legacy (non-coalescing) scheme.

- RSC is enabled in the destination receive queue by the RSCCTL.*RSCEN*. In this case, software must set the SRRCTL.*DESCTYPE* field in the relevant queues to advanced descriptor modes.
- RSC is further enabled by the RSCINT.*RSCEN* for the receive queues associated to the interrupts defined by the RSCINT registers.
- The SRRCTL[n].*BSIZEHEADER* (header buffer size) must be larger than the packet header (even if header split is not enabled). A minimum size of 128 bytes for the header buffer addresses this requirement.
- The received packet has no MAC errors and no TCP/IP checksum errors. MAC errors are:
  - CRC error.
  - Undersize frame received.
  - Oversize frame received.
  - Error control byte received in mid-packet.
  - Illegal code byte received in mid-packet.
- If the *Length* field in the IP header does not cover the entire packet (as the case for padding bytes), the received packet is not a candidate for RSC.
- The packet type is TCP/IPv4 (non-SNAP) with optional VLAN header(s). RSC is not supported for IPv6 packets.
- The packet is not an NVGRE or VXLAN packet.
- IP header does not carry any option headers.
- See Section 4.6.7.2 for the required configuration to enable or disable NFS coalescing.
- The TCP segment is not fragmented.
- The following TCP flags are inactive: FIN, SYN, RST, URG, ECE, CWR, NS and the other three reserved TCP flags (see TCP Flags mapping in Table 7-72).
- The *ECT* and *CE* bits in the *TOS* field in the IP header are not equal to 11b (see the flags in Table 7-73).
- Packets with PSH TCP flag are coalesced but also close the large receive.
- The packet does not carry any TCP option headers.
- RSC is not supported for a switched packet transmitted from a local VM.
- When a Rx packet is replicated or mirrored, it can be coalesced only on the Rx queue that belongs to the source VM.
- Note that there are no limitations on the maximum packet length including jumbo packets.
- If there is already an active RSC for the matched flow, a few additional conditions should be met as listed in Section 7.10.4.



The supported packet format is as follows:

Table 7-71 Packet Fields

Size	Packet Fields
6 bytes	Destination Ethernet MAC Address.
6 bytes	Source Ethernet MAC Address.
[4/6/8 bytes]	Optional External Tag (VLAN or E-tag).
[4 bytes]	Optional VLAN.
2 bytes	Ethernet type field equals 0x0800 (MS byte first on the wire).
20 bytes	IPv4 header with no options.
20 bytes	Basic TCP header (no options — refer to the rows that follow).
[10 bytes]	Optional TCP time stamp header: 1 byte Kind 0x08 1 byte Length 0x0A 4 bytes TS value variable 4 bytes TS echo reply variable
[1 byte]	Optional TCP no operation header: 1 byte Kind 0x01
[1 byte]	Optional TCP end of option header list: 1 byte Kind 0x00
Variable length	TCP payload (RSC candidate must have payload size greater than zero).

Table 7-72 Packet Format Supported by RSC

11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN

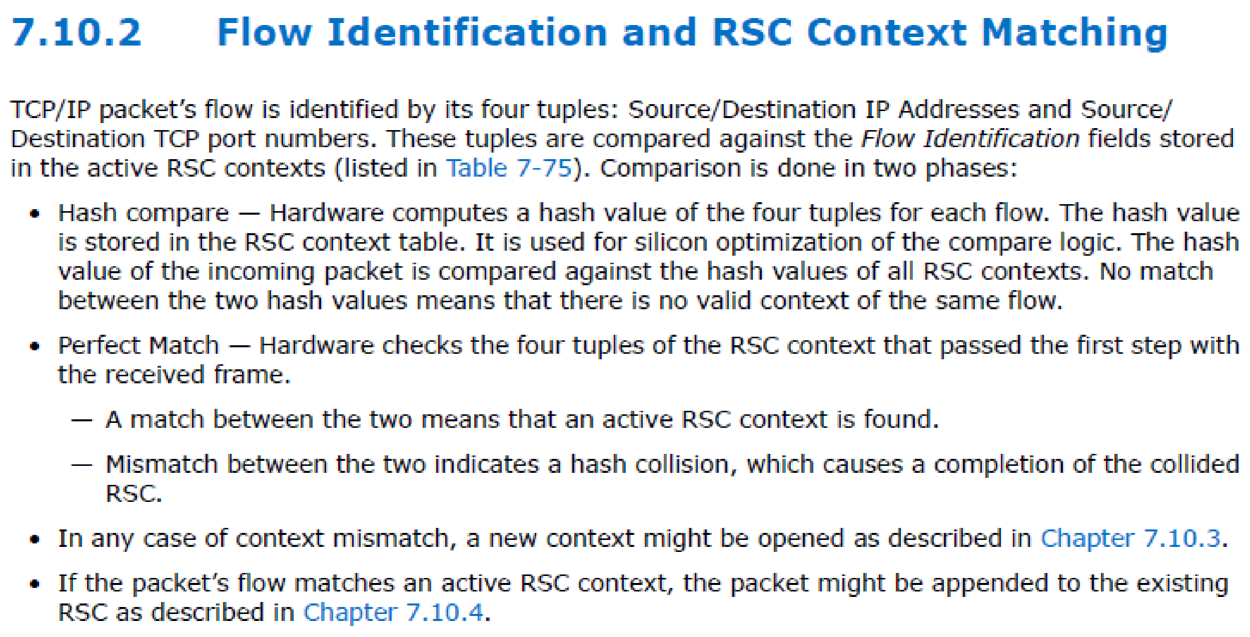
Table 7-73 IP TOS Field — Bit Map

7	6	5	4	3	2	1	0
TOS (DS)						ECT	CE

Table 7-74 TCP Time-Stamp Option Header (RFC 1323)

1 Byte: First on the Wire	1 Byte	4 Bytes	4 Bytes: Last on the Wire
Kind = 0x8	Length = 10	TS Value (TSval)	TS Echo Reply (TSecr)

1(c) *generating a flow key to identify a first communication flow that includes said first packet, wherein said flow key includes a TCP connection for the communication flow;* – In connection with the RSC functionality discussed above, the NM10GR Card of the SV7220G2 Server generates a flow key to identify a first communication flow that includes the incoming packet. The NM10GR Card computes a hash value from information included in the header of the received packet, including the IP destination address, IP source address, TCP destination port, and TCP source port for the identified communication flow, and compares that hash value to active RSC contexts present on the NM10GR Card for purposes of flow identification, as set forth, for example, in Section 7.10.2 and Figure 7-43 of the X550 Datasheet. The NM10GR Card will create a new RSC context that includes the received packet if the Controller determines that conditions exist that warrant opening a new RSC context. As shown in Table 7-75 of the X550 Datasheet, for example, the RSC Context includes the Flow Identification for the TCP connection, including the IP destination address, IP source address, TCP destination port, and TCP source port for the identified communication flow.



7.10.2 Flow Identification and RSC Context Matching

TCP/IP packet's flow is identified by its four tuples: Source/Destination IP Addresses and Source/Destination TCP port numbers. These tuples are compared against the *Flow Identification* fields stored in the active RSC contexts (listed in Table 7-75). Comparison is done in two phases:

- Hash compare — Hardware computes a hash value of the four tuples for each flow. The hash value is stored in the RSC context table. It is used for silicon optimization of the compare logic. The hash value of the incoming packet is compared against the hash values of all RSC contexts. No match between the two hash values means that there is no valid context of the same flow.
- Perfect Match — Hardware checks the four tuples of the RSC context that passed the first step with the received frame.
  - A match between the two means that an active RSC context is found.
  - Mismatch between the two indicates a hash collision, which causes a completion of the collided RSC.
- In any case of context mismatch, a new context might be opened as described in Chapter 7.10.3.
- If the packet's flow matches an active RSC context, the packet might be appended to the existing RSC as described in Chapter 7.10.4.

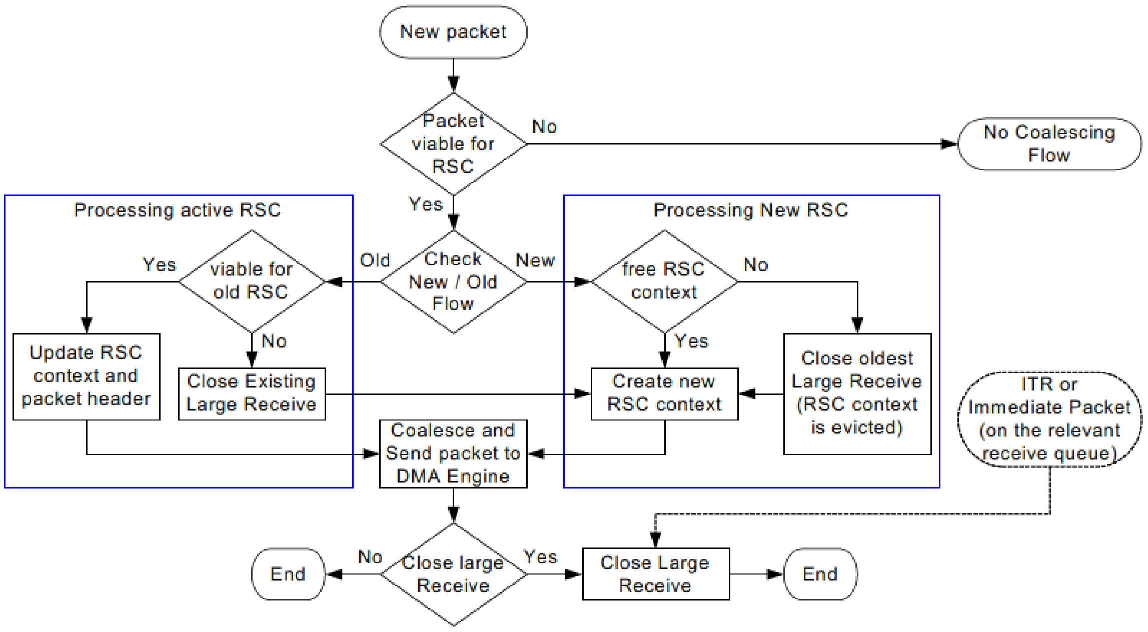


Figure 7-43 RSC Event Flow

Table 7-75 RSC Context

Size	Name	Description
		Flow Identification[1]
1 bit	CVALID	Context valid indication. Set to 1b by hardware when a new context is defined. Cleared to 0b when RSC completes.
1 byte	CHASH	Context hash value (logic XOR of all bytes of the four tuples).
16 bytes	IPDADDR	IP destination address (set to zero for inactive context).
16 bytes	IPSADDR	IP source address (set to zero for inactive context).
1 bit	IP4TYPE	Defines IP version type (set to 1b for IPv4).
2 bytes	TCPDPORT	TCP destination port.
2 bytes	TCPSPORT	TCP source port.

1(d) *associating an operation code with said first packet, wherein said operation code indicates a status of said first packet, including whether said packet is a candidate for transfer to the host computer system that avoids processing said header portion by the host computer system in accordance with said TCP protocol; and* – In connection with the RSC functionality discussed above, the the NM10GR Card of the SV7220G2 Server associates an operation code with the packet that indicates a status of the packet, including whether the packet is a viable candidate for RSC coalescing, as shown, for example, in Figure 7-43 of the X550 Datasheet.

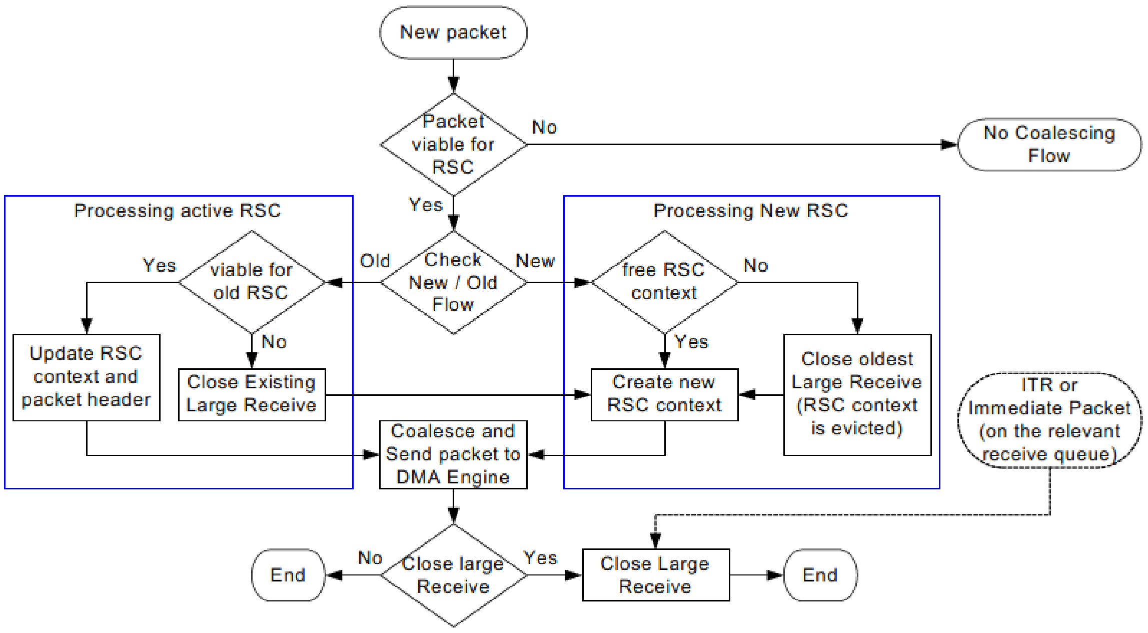


Figure 7-43 RSC Event Flow

If a packet is viable for RSC coalescing, the packet is a candidate for transfer to the host computer system of the SV7220G2 Server that avoids processing by the host of the header portion of the packet in accordance with the TCP protocol, as explained in Section 7.10, and shown in Figures 7-41 and 7-42 (and elsewhere) of the X550 Datasheet.

7.10 Receive Side Coalescing (RSC)

The X550 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X550 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

The X550 digests received packets and categorizes them by their TCP/IP connections (flows). For each flow, hardware coalesces the packets as shown in Figure 7-41 and Figure 7-42 (the colored parameters are explained in the RSC context table and receive descriptor sections). The X550 can handle up to 32 concurrent flows per LAN port at any given time. Each flow handled by RSC offload has an associated context. The X550 opens and closes the RSC contexts autonomously with no need for any software intervention. Software needs only to enable RSC in the selected receive queues.

Note: When RSC is enabled, advanced receive descriptors should be used and CRC strip should be enabled.

Figure 7-41 shows a top level flow diagram that is used for RSC functionality. The following sections provide a detailed explanation of this flow as well as the memory structures and device settings that support the RSC functionality.

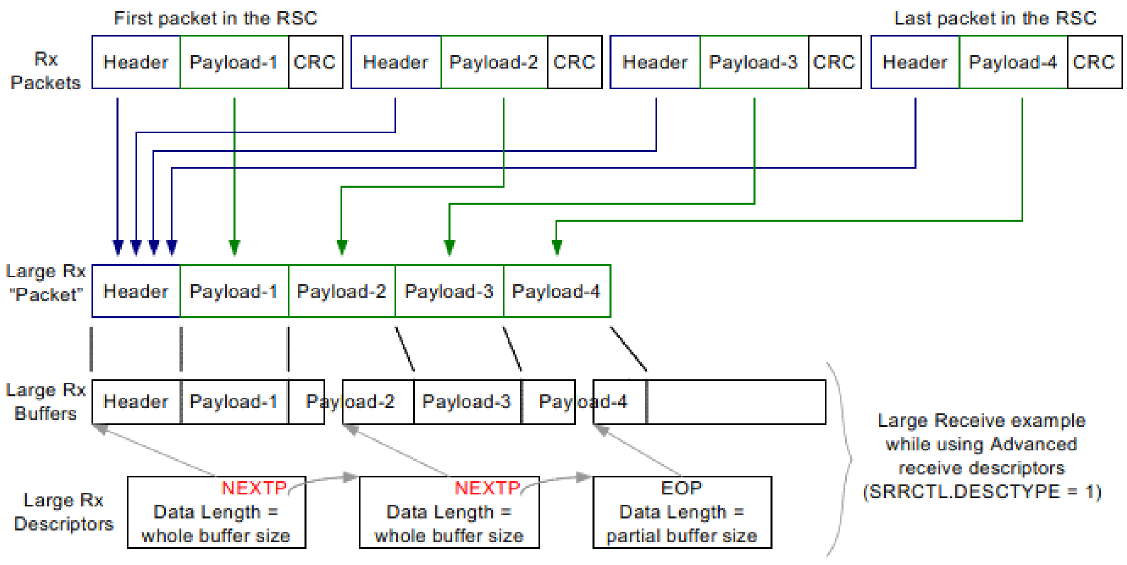


Figure 7-41 RSC Functionality (No Header Split)

***

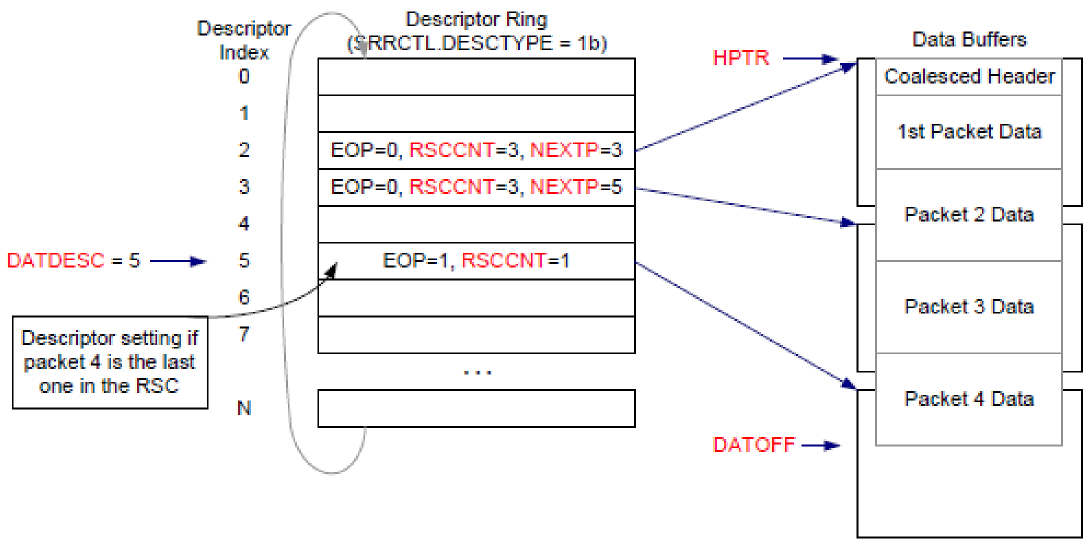


Figure 7-42 RSC Functionality (No Header Split)

1(e) *processing, by the network interface, said packet according to the TCP connection, including updating a control block representing the TCP connection on the network interface.* – In connection with the RSC functionality discussed above, the network interface of the SV7220G2 Server incorporating the NM10GR Card processes the received packets according to the TCP connection, including updating a control block representing the TCP connection. For example, as part of processing of coalesced packets using the RSC functionality described above, the NM10GR Card dynamically updates data in the RSC context, as described in the X550 Datasheet, for example, in Table 7-75.

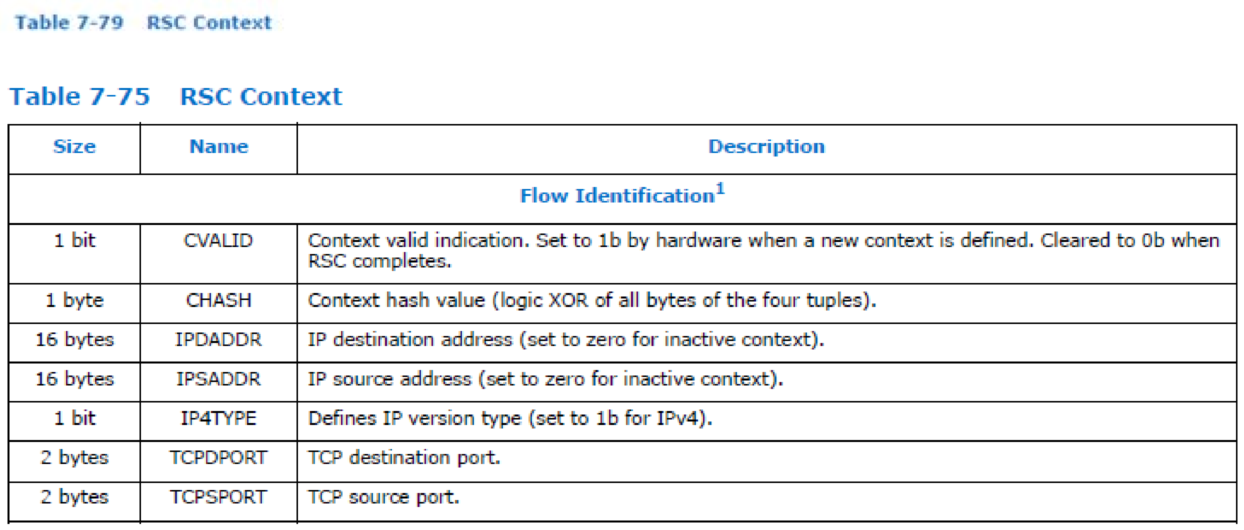


Table 7-79 RSC Context

Table 7-75 RSC Context

Size	Name	Description
Flow Identification[1]		
1 bit	CVALID	Context valid indication. Set to 1b by hardware when a new context is defined. Cleared to 0b when RSC completes.
1 byte	CHASH	Context hash value (logic XOR of all bytes of the four tuples).
16 bytes	IPDADDR	IP destination address (set to zero for inactive context).
16 bytes	IPSADDR	IP source address (set to zero for inactive context).
1 bit	IP4TYPE	Defines IP version type (set to 1b for IPv4).
2 bytes	TCPDPORT	TCP destination port.
2 bytes	TCPSPORT	TCP source port.

94. As of no later than the filing and service of this Complaint, Defendants are also indirectly infringing the '880 patent.

95. Defendants have actual knowledge of Alacritech's rights in the '880 patent and details of Defendants' infringement of the '880 patent based on at least the filing and service of this Complaint.

96. Defendants make, use, import, offer for sale, and/or sell the '880 Accused Products (including, for example, the products described above) with knowledge of or willful blindness to the fact that their actions will induce their customers to infringe the '880 patent. Defendants induce others to infringe the '880 patent in violation of 35 U.S.C. § 271 by encouraging and facilitating others to practice the '880 patent's inventions for accelerated network communications with intent that those performing the acts infringe the '880 patent. For example, Defendants sell and/or provide '880 Accused Products that perform infringing RSC

functionality by default. When Defendants' customers use the '880 Accused Products for their intended purpose, they practice one or more claims of the '880 patent. Defendants instructs their customers how to set up and use '880 Accused Products. In some cases, Defendants even sets up '880 Accused Products for their customers. Once they have been set up, Defendants' customers infringe one or more claims of the '880 patent simply by using the '880 Accused Products. By selling and/or providing their customers '880 Accused Products that infringe by default when used for their intended purpose, and by setting up for their customers and/or instructing their customers how to set up and use those products, Defendants induce those customers to infringe the '880 patent.

97. Defendants also contribute to the infringement of the '880 patent in violation of 35 U.S.C. § 271. Defendants know that infringing components of the '880 Accused Products are especially made or especially adapted for use in the infringement of the '880 patent. The infringing components of these products are not staple articles or commodities of commerce suitable for substantial non-infringing use, and the infringing components of these products are a material part of the invention of the '880 patent. The '880 Accused Products contain infringing components such as specialized hardware and/or software for performing infringing RSC functionality, which are separable from the '880 Accused Products, material to practicing the '880 patent's inventions for accelerated network communications, and have no substantial non-infringing use. Accordingly, Defendants are also contributing to the direct infringement of the '880 patent by their customers.

98. Defendants are not licensed or otherwise authorized by Alacritech to practice, contributorily practice and/or induce third parties to practice the claims of the '880 patent.

99. By reason of Defendants' infringing activities, Alacritech has suffered, and will continue to suffer, substantial damages in an amount to be proven at trial. Alacritech is entitled to a money judgment in an amount adequate to compensate for Defendants' infringement, but in no event less than a reasonable royalty for the use made of the invention by Defendants, together with interest and costs as fixed by the Court.

100. Defendants' continuing acts of infringement have caused and will continue to cause Alacritech irreparable harm, for which Alacritech has no adequate remedy at law, unless Defendants' continuing acts of infringement are enjoined by the Court. The hardships that an injunction would impose are less than those faced by Alacritech should an injunction not issue. The public interest would be served by issuance of an injunction.

101. Defendants' infringement of the '880 patent is exceptional and entitles Alacritech to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

COUNT VI

INFRINGEMENT OF U.S. PATENT NO. 8,805,948

102. Alacritech re-alleges and incorporates by reference each of the allegations of paragraphs 1-36 set forth above as though fully set forth herein.

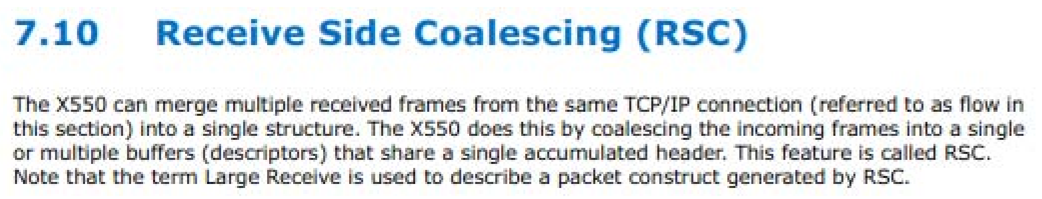
103. Alacritech is the current exclusive owner and assignee of all right, title, and interest in and to U.S. Patent No. 8,805,948 (the "'948 patent"), titled "Intelligent Network Interface System and Method for Protocol Processing," duly and legally issued by the United States Patent and Trademark Office on August 12, 2014, including the right to bring this suit for injunctive relief and damages. A true and correct copy of the '948 patent is attached hereto as Exhibit F.

104. The '948 patent is valid and enforceable.

105. Defendants have directly infringed and are currently directly infringing the '948 patent by making, using, selling, offering for sale, and/or importing into the United States, without authority, products, methods, equipment, and/or services that practice one or more claims of the '948 patent in connection with infringing RSC functionality, including but not limited to the Wiwynn SV100G2, SV300, SV300G2, SV320, SV320G2, SV324G2, SV5270G2-R, SV5270G2-S, SV7110, SV7220G2-N, SV7220G2-P, SV7220G2-S, SV7220G2-V, NM10GR, and NM10GS; any other activities, products and/or services involving the products identified above; and any other activities, products and/or services that practice and/or support similarly infringing RSC functionality (collectively, "the '948 Accused Products"). The '948 Accused Products are non-limiting examples that were identified based on publicly available information, and Alacritech reserves the right to identify additional infringing activities, products and services, including, for example, on the basis of information obtained during discovery.

106. As just one non-limiting example, set forth below (with claim language in italics) is a description of Defendants' infringement of exemplary claim 17 of the '948 patent in connection with the Wiwynn SV7220G2 Server configured with a Wiwynn NM10GR Mezzanine Card. This description is based on publicly available information. Alacritech reserves the right to modify this description, including, for example, on the basis of information about the '948 Accused Products that it obtains during discovery.

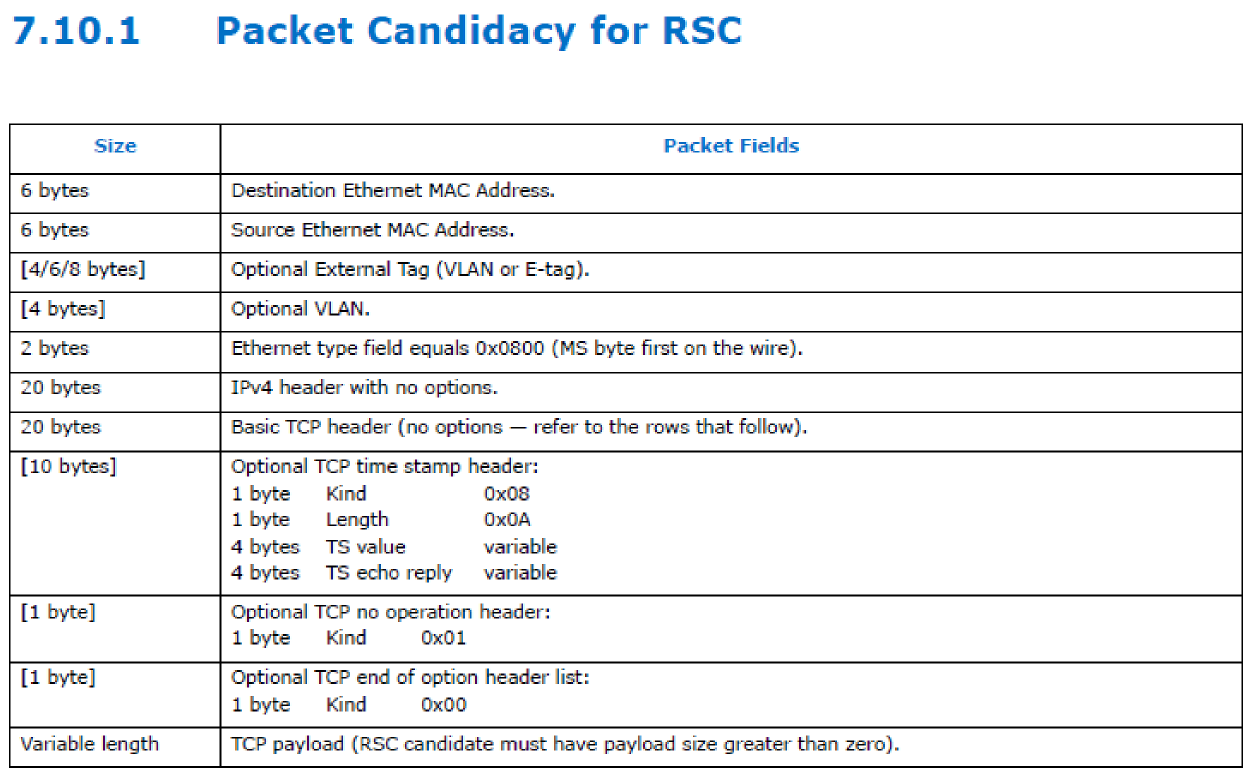
17(a) *An apparatus for network communication, the apparatus comprising: a host computer running a protocol stack including an Internet Protocol (IP) layer and a Transmission Control Protocol (TCP) layer, the protocol stack adapted to establish a TCP connection for an application layer running above the TCP layer, the TCP connection being defined by source and destination IP addresses and source and destination TCP ports* - Defendants make, use, sell, offer for sale, and/or import servers, network interface devices (e.g., controllers and cards), and other network products that support infringing RSC functionality. One example of an infringing products is the Wiwynn SV7220G2 Server (the "SV7220G2 Server") configured with a Wiwynn NM10GR Mezzanine Card (the "NM10GR Card") comprising an Intel X550 controller (the "X550 Controller"). *See, e.g.*, "Wiwynn 2016 OCP US Summit Product Brochure," available at https://www.circleb.eu/wp-content/uploads/2016/04/OCP-US-Summit-OPS.pdf ("OCP Product Brochure"); "Wiwynn SV7220G2 Series Datasheet," available at http://www.wiwynn.com/usr_files/20150413140223_Datasheet%20-%20SV7220G2_150413.pdf ("SV7220G2 Datasheet"). A SV7220G2 Server comprises a host computer that runs a protocol stack with TCP and IP layers (e.g., the TCP/IP stack of the server's Microsoft Windows or Linux operating system) that establishes a TCP connection for an application layer running above the TCP layer, defined by source and destination IP addresses and TCP ports. *See, e.g.*, OCP Product Brochure; SV7220G2 Datasheet. The "Intel Ethernet Controller X550 Datasheet," Revision 2.1 (May 2016), available at http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/ethernet-x550-datasheet.pdf (the "X550 Datasheet") provides an overview of the X550 Controller and explains how it interacts with a compatible server (such as the SV7220G2 Server), including, for example, describing in Section 7.10 how it performs RSC functionality associated with processing incoming data packets. Section 7.10.1 of the X550 Datasheet describes the structure of a packet, which contains information (e.g., destination IP addresses and TCP ports) corresponding to the TCP connection to which the packet pertains.



7.10 Receive Side Coalescing (RSC)

The X550 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X550 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

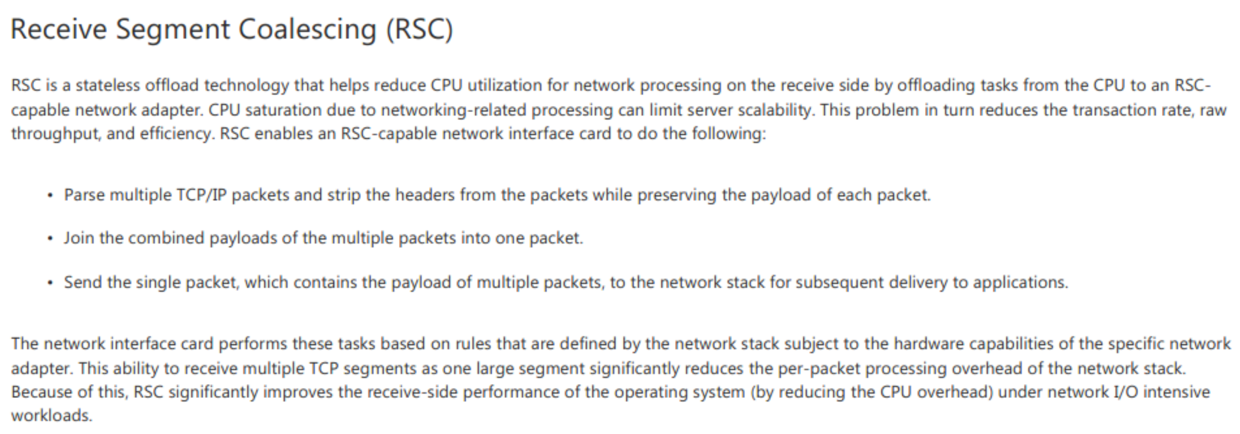
***



7.10.1 Packet Candidacy for RSC

Size	Packet Fields
6 bytes	Destination Ethernet MAC Address.
6 bytes	Source Ethernet MAC Address.
[4/6/8 bytes]	Optional External Tag (VLAN or E-tag).
[4 bytes]	Optional VLAN.
2 bytes	Ethernet type field equals 0x0800 (MS byte first on the wire).
20 bytes	IPv4 header with no options.
20 bytes	Basic TCP header (no options — refer to the rows that follow).
[10 bytes]	Optional TCP time stamp header: 1 byte Kind 0x08 1 byte Length 0x0A 4 bytes TS value variable 4 bytes TS echo reply variable
[1 byte]	Optional TCP no operation header: 1 byte Kind 0x01
[1 byte]	Optional TCP end of option header list: 1 byte Kind 0x00
Variable length	TCP payload (RSC candidate must have payload size greater than zero).

Microsoft describes how a network interface device supporting infringing RSC functionality reduces communications processing by the TCP/IP stack on a server (such as the SV7220G2 Server incorporating the NM10GR Card) running the Microsoft Windows Server Operating System, which enables RSC by default. *See*, *e.g.*, "Receive Segment Coalescing," available at https://technet.microsoft.com/en-us/library/hh997024.aspx.



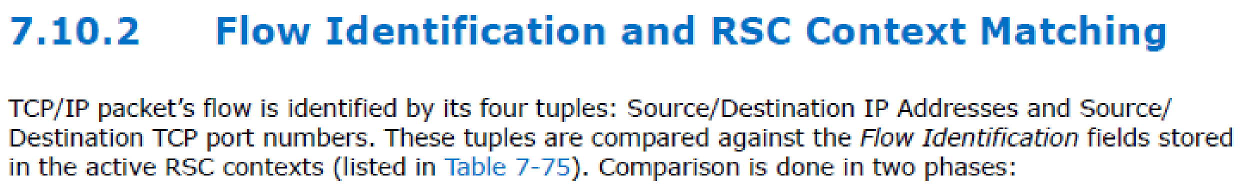
Receive Segment Coalescing (RSC)

RSC is a stateless offload technology that helps reduce CPU utilization for network processing on the receive side by offloading tasks from the CPU to an RSC-capable network adapter. CPU saturation due to networking-related processing can limit server scalability. This problem in turn reduces the transaction rate, raw throughput, and efficiency. RSC enables an RSC-capable network interface card to do the following:

- Parse multiple TCP/IP packets and strip the headers from the packets while preserving the payload of each packet.
- Join the combined payloads of the multiple packets into one packet.
- Send the single packet, which contains the payload of multiple packets, to the network stack for subsequent delivery to applications.

The network interface card performs these tasks based on rules that are defined by the network stack subject to the hardware capabilities of the specific network adapter. This ability to receive multiple TCP segments as one large segment significantly reduces the per-packet processing overhead of the network stack. Because of this, RSC significantly improves the receive-side performance of the operating system (by reducing the CPU overhead) under network I/O intensive workloads.

Section 7.10.2 and Table 7-75 of the X550 Datasheet, for example, describe how RSC functionality relies on the source and destination IP addresses and TCP ports defining the TCP connection established by the TCP/IP stack.

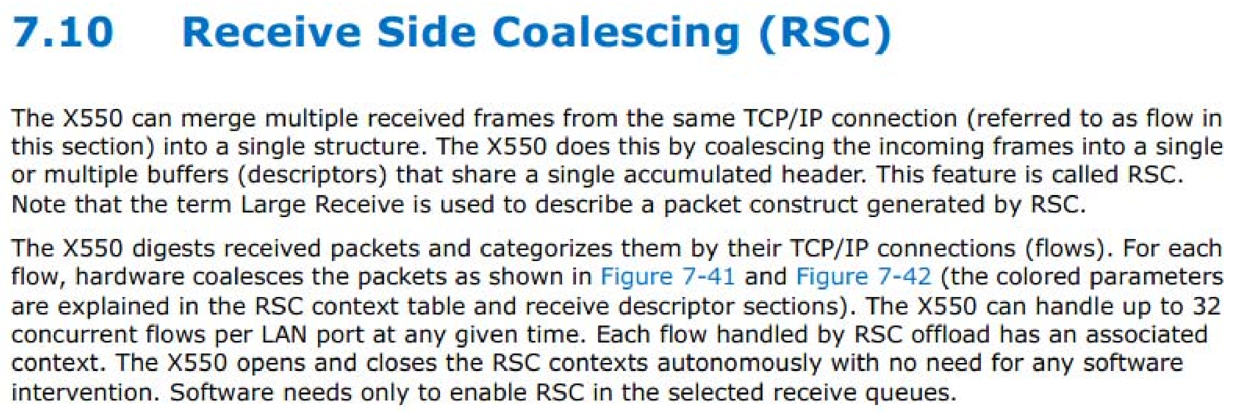


7.10.2 Flow Identification and RSC Context Matching

TCP/IP packet's flow is identified by its four tuples: Source/Destination IP Addresses and Source/Destination TCP port numbers. These tuples are compared against the *Flow Identification* fields stored in the active RSC contexts (listed in Table 7-75). Comparison is done in two phases:

17(b) *a network interface that is connected to the host computer by an input/output bus, the network interface adapted to parse the headers of received packets to determine whether the headers have the IP addresses and TCP ports that define the TCP connection and to check whether the packets have certain exception conditions, including whether the packets are IP fragmented, have a FIN flag set, or are out of order, the network interface having logic that directs any of the received packets that have the exception conditions to the protocol stack for processing, and directs the received packets that do not have any of the exception conditions to have their headers removed and their payload data stored together in a buffer of the host computer, such that the payload data is stored in the buffer in order and without any TCP header stored between the payload data that came from different packets of the received packets.* - In connection with the RSC functionality discussed above, the SV7220G2 Server incorporating the

NM10GR Card includes a network interface comprising the NM10GR Card with its X550 Controller, connected to the host computer by an I/O bus and adapted to parse the headers of received packets to determine whether the headers have the IP addresses and TCP ports defining the TCP connection, as explained, for example, in the Section 7.10 introduction and in Section 7.10.2 of the X550 Datasheet.

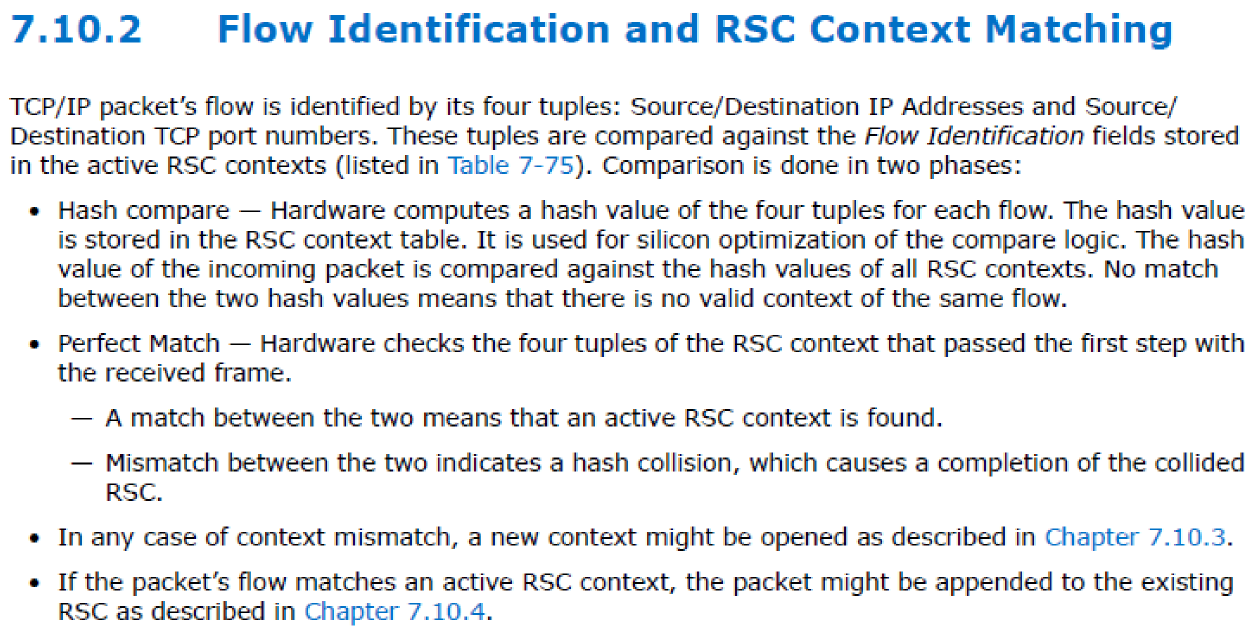


7.10 Receive Side Coalescing (RSC)

The X550 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X550 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

The X550 digests received packets and categorizes them by their TCP/IP connections (flows). For each flow, hardware coalesces the packets as shown in Figure 7-41 and Figure 7-42 (the colored parameters are explained in the RSC context table and receive descriptor sections). The X550 can handle up to 32 concurrent flows per LAN port at any given time. Each flow handled by RSC offload has an associated context. The X550 opens and closes the RSC contexts autonomously with no need for any software intervention. Software needs only to enable RSC in the selected receive queues.

***

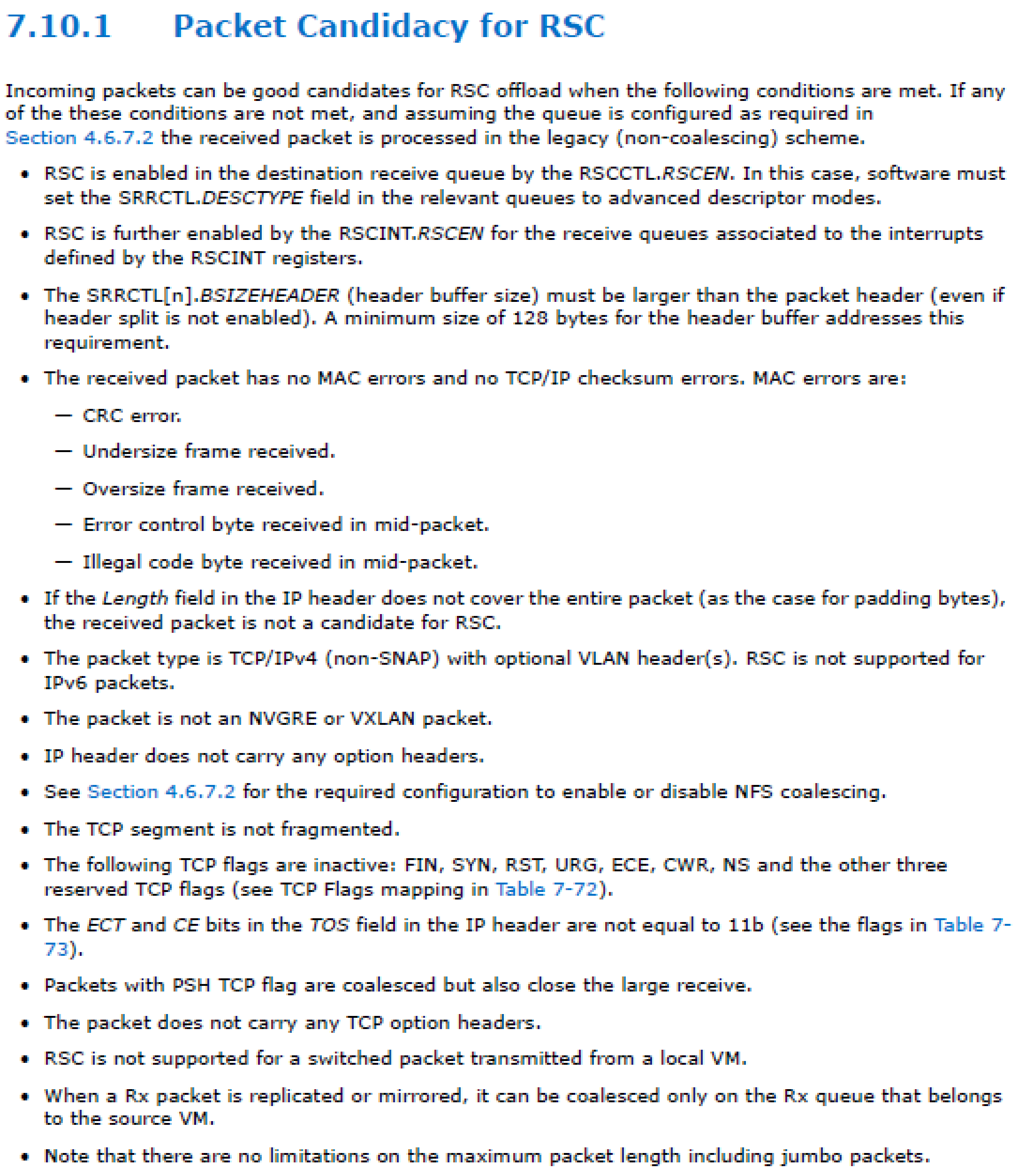


7.10.2 Flow Identification and RSC Context Matching

TCP/IP packet's flow is identified by its four tuples: Source/Destination IP Addresses and Source/ Destination TCP port numbers. These tuples are compared against the *Flow Identification* fields stored in the active RSC contexts (listed in Table 7-75). Comparison is done in two phases:

- Hash compare — Hardware computes a hash value of the four tuples for each flow. The hash value is stored in the RSC context table. It is used for silicon optimization of the compare logic. The hash value of the incoming packet is compared against the hash values of all RSC contexts. No match between the two hash values means that there is no valid context of the same flow.
- Perfect Match — Hardware checks the four tuples of the RSC context that passed the first step with the received frame.
  - A match between the two means that an active RSC context is found.
  - Mismatch between the two indicates a hash collision, which causes a completion of the collided RSC.
- In any case of context mismatch, a new context might be opened as described in Chapter 7.10.3.
- If the packet's flow matches an active RSC context, the packet might be appended to the existing RSC as described in Chapter 7.10.4.

The NM10GR Card also checks whether the packets have certain exceptions, including IP fragmented packets, FIN flag set, or certain out-of-order exceptions, to determine if a packet is viable for RSC, as shown in sections 7.10.1 and 7.10.4 (and elsewhere) of the X550 Datasheet.

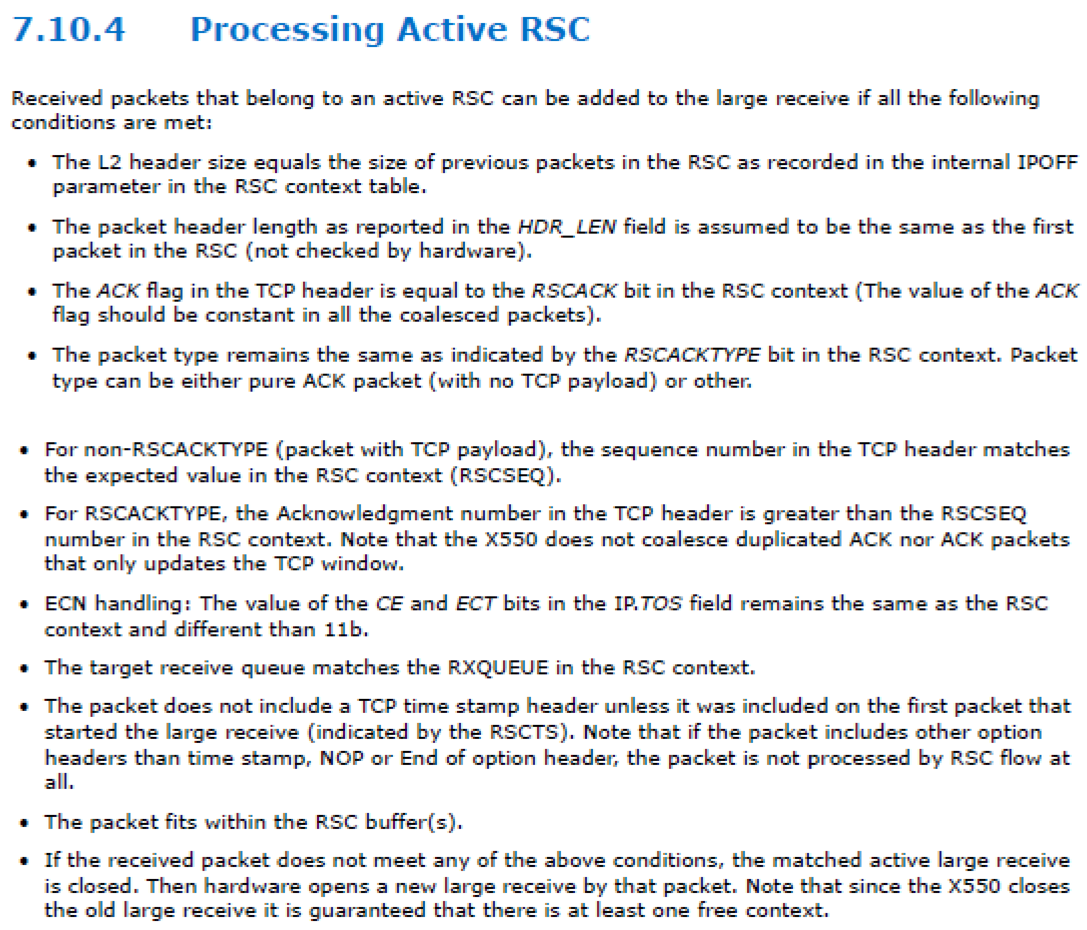


7.10.1 Packet Candidacy for RSC

Incoming packets can be good candidates for RSC offload when the following conditions are met. If any of the these conditions are not met, and assuming the queue is configured as required in Section 4.6.7.2 the received packet is processed in the legacy (non-coalescing) scheme.

- RSC is enabled in the destination receive queue by the RSCCTL.*RSCEN*. In this case, software must set the SRRCTL.*DESCTYPE* field in the relevant queues to advanced descriptor modes.
- RSC is further enabled by the RSCINT.*RSCEN* for the receive queues associated to the interrupts defined by the RSCINT registers.
- The SRRCTL[n].*BSIZEHEADER* (header buffer size) must be larger than the packet header (even if header split is not enabled). A minimum size of 128 bytes for the header buffer addresses this requirement.
- The received packet has no MAC errors and no TCP/IP checksum errors. MAC errors are:
  — CRC error.
  — Undersize frame received.
  — Oversize frame received.
  — Error control byte received in mid-packet.
  — Illegal code byte received in mid-packet.
- If the *Length* field in the IP header does not cover the entire packet (as the case for padding bytes), the received packet is not a candidate for RSC.
- The packet type is TCP/IPv4 (non-SNAP) with optional VLAN header(s). RSC is not supported for IPv6 packets.
- The packet is not an NVGRE or VXLAN packet.
- IP header does not carry any option headers.
- See Section 4.6.7.2 for the required configuration to enable or disable NFS coalescing.
- The TCP segment is not fragmented.
- The following TCP flags are inactive: FIN, SYN, RST, URG, ECE, CWR, NS and the other three reserved TCP flags (see TCP Flags mapping in Table 7-72).
- The *ECT* and *CE* bits in the *TOS* field in the IP header are not equal to 11b (see the flags in Table 7-73).
- Packets with PSH TCP flag are coalesced but also close the large receive.
- The packet does not carry any TCP option headers.
- RSC is not supported for a switched packet transmitted from a local VM.
- When a Rx packet is replicated or mirrored, it can be coalesced only on the Rx queue that belongs to the source VM.
- Note that there are no limitations on the maximum packet length including jumbo packets.

***



7.10.4 Processing Active RSC

Received packets that belong to an active RSC can be added to the large receive if all the following conditions are met:

- The L2 header size equals the size of previous packets in the RSC as recorded in the internal IPOFF parameter in the RSC context table.
- The packet header length as reported in the *HDR_LEN* field is assumed to be the same as the first packet in the RSC (not checked by hardware).
- The *ACK* flag in the TCP header is equal to the *RSCACK* bit in the RSC context (The value of the *ACK* flag should be constant in all the coalesced packets).
- The packet type remains the same as indicated by the *RSCACKTYPE* bit in the RSC context. Packet type can be either pure ACK packet (with no TCP payload) or other.

- For non-RSCACKTYPE (packet with TCP payload), the sequence number in the TCP header matches the expected value in the RSC context (RSCSEQ).
- For RSCACKTYPE, the Acknowledgment number in the TCP header is greater than the RSCSEQ number in the RSC context. Note that the X550 does not coalesce duplicated ACK nor ACK packets that only updates the TCP window.
- ECN handling: The value of the *CE* and *ECT* bits in the IP.*TOS* field remains the same as the RSC context and different than 11b.
- The target receive queue matches the RXQUEUE in the RSC context.
- The packet does not include a TCP time stamp header unless it was included on the first packet that started the large receive (indicated by the RSCTS). Note that if the packet includes other option headers than time stamp, NOP or End of option header, the packet is not processed by RSC flow at all.
- The packet fits within the RSC buffer(s).
- If the received packet does not meet any of the above conditions, the matched active large receive is closed. Then hardware opens a new large receive by that packet. Note that since the X550 closes the old large receive it is guaranteed that there is at least one free context.

The NM10GR Card has logic that directs any of the received packets that have the exception conditions to the protocol stack for conventional processing, and directs received packets that do not have any of the exception conditions to have their headers removed and their payload data stored together in a host-computer buffer, in order, and without any TCP header stored between the payload data that came from the different received packets. This is described in the sections above and is also shown, for example, in figures 7-41, 7-42, and 7-43 of the X550 Datasheet.

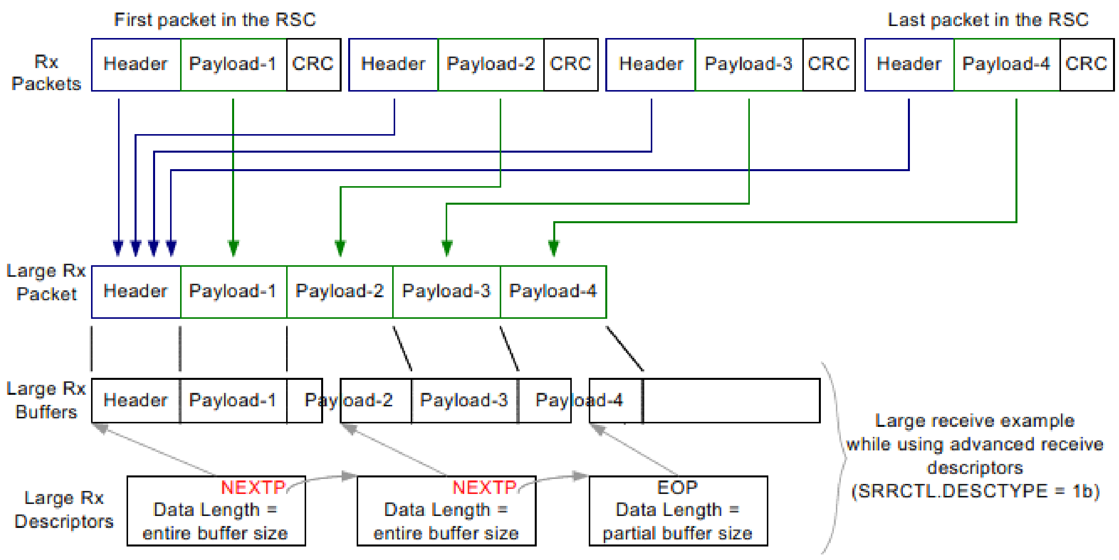


Figure 7-41 RSC Functionality (No Header Split)

***

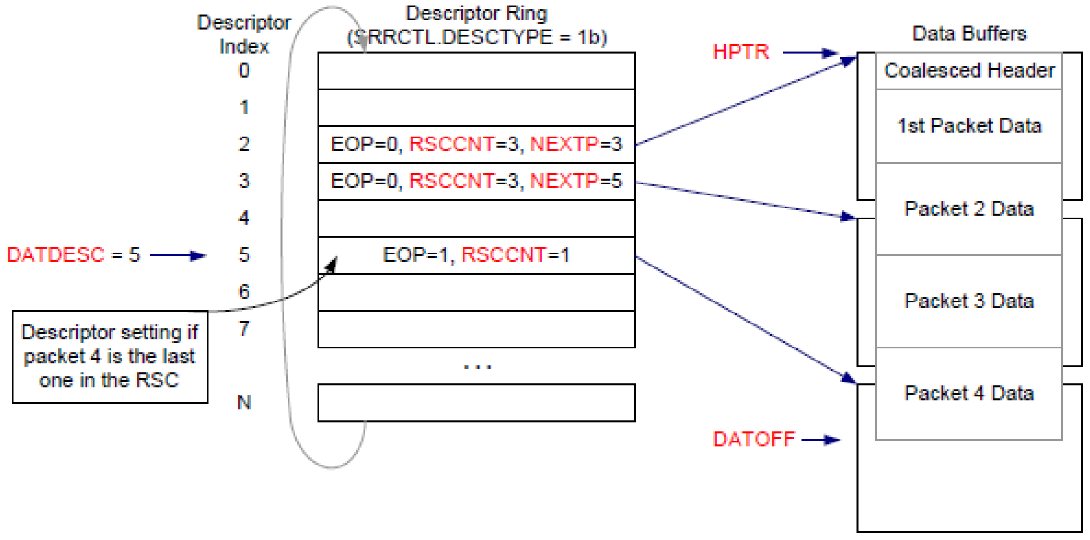


Figure 7-42 RSC Functionality (No Header Split)

***

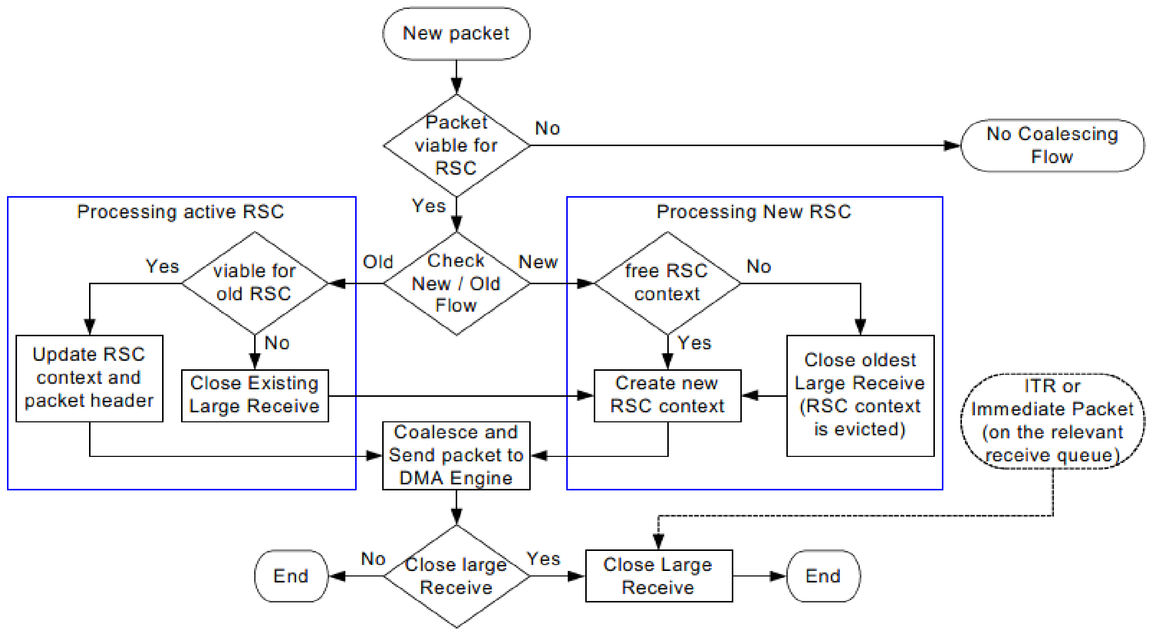


Figure 7-43 RSC Event Flow

107. As of no later than the filing and service of this Complaint, Defendants are also indirectly infringing the '948 patent.

108. Defendants have actual knowledge of Alacritech's rights in the '948 patent and details of Defendants' infringement of the '948 patent based on at least the filing and service of this Complaint.

109. Defendants make, use, import, offer for sale, and/or sell the '948 Accused Products (including, for example, the products described above) with knowledge of or willful blindness to the fact that their actions will induce their customers to infringe the '948 patent. Defendants induce others to infringe the '948 patent in violation of 35 U.S.C. § 271 by encouraging and facilitating others to practice the '948 patent's inventions for accelerated network communications with intent that those performing the acts infringe the '948 patent. For example, Defendants sell and/or provide '948 Accused Products that perform infringing RSC functionality by default. When Defendants' customers use the '948 Accused Products for their intended purpose, they practice one or more claims of the '948 patent. Defendants instructs their

customers how to set up and use '948 Accused Products. In some cases, Defendants even sets up '948 Accused Products for their customers. Once they have been set up, Defendants' customers infringe one or more claims of the '948 patent simply by using the '948 Accused Products. By selling and/or providing their customers '948 Accused Products that infringe by default when used for their intended purpose, and by setting up for their customers and/or instructing their customers how to set up and use those products, Defendants induce those customers to infringe the '948 patent.

110. Defendants also contribute to the infringement of the '948 patent in violation of 35 U.S.C. § 271. Defendants know that infringing components of the '948 Accused Products are especially made or especially adapted for use in the infringement of the '948 patent. The infringing components of these products are not staple articles or commodities of commerce suitable for substantial non-infringing use, and the infringing components of these products are a material part of the invention of the '948 patent. The '948 Accused Products contain infringing components such as specialized hardware and/or software for performing infringing RSC functionality, which are separable from the '948 Accused Products, material to practicing the '948 patent's inventions for accelerated network communications, and have no substantial non-infringing use. Accordingly, Defendants are also contributing to the direct infringement of the '948 patent by their customers.

111. Defendants are not licensed or otherwise authorized by Alacritech to practice, contributorily practice and/or induce third parties to practice the claims of the '948 patent.

112. By reason of Defendants' infringing activities, Alacritech has suffered, and will continue to suffer, substantial damages in an amount to be proven at trial. Alacritech is entitled to a money judgment in an amount adequate to compensate for Defendants' infringement, but in

no event less than a reasonable royalty for the use made of the invention by Defendants, together with interest and costs as fixed by the Court.

113. Defendants' continuing acts of infringement have caused and will continue to cause Alacritech irreparable harm, for which Alacritech has no adequate remedy at law, unless Defendants' continuing acts of infringement are enjoined by the Court. The hardships that an injunction would impose are less than those faced by Alacritech should an injunction not issue. The public interest would be served by issuance of an injunction.

114. Defendants' infringement of the '948 patent is exceptional and entitles Alacritech to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

COUNT VII

INFRINGEMENT OF U.S. PATENT NO. 9,055,104

115. Alacritech re-alleges and incorporates by reference each of the allegations of paragraphs 1-36 set forth above as though fully set forth herein.

116. Alacritech is the current exclusive owner and assignee of all right, title, and interest in and to U.S. Patent No. 9,055,104 (the "'104 patent"), titled "Freeing Transit Memory on a Network Interface Device prior to Receiving an Acknowledgement that Transmit Data Has Been Received by a Remote Device," duly and legally issued by the United States Patent and Trademark Office on June 9, 2015, including the right to bring this suit for injunctive relief and damages. A true and correct copy of the '104 patent is attached hereto as Exhibit G.

117. The '104 patent is valid and enforceable.

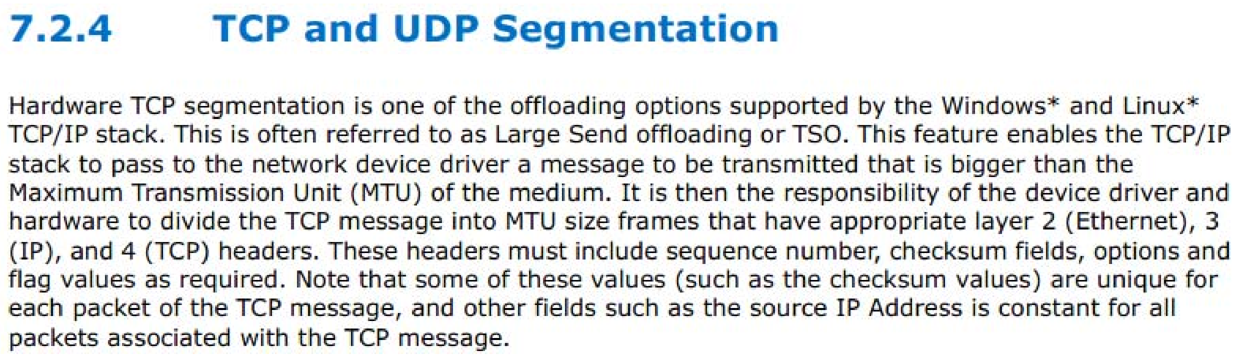
118. Defendants have directly infringed and are currently directly infringing the '104 patent by making, using, selling, offering for sale, and/or importing into the United States, without authority, products, methods, equipment, and/or services that practice one or more claims of the '104 patent in connection with infringing LSO functionality, including but not

limited to the Wiwynn SV100G2, SV300, SV300G2, SV320, SV320G2, SV324G2, SV5270G2-R, SV5270G2-S, SV7110, SV7220-2A, SV7220-2P, SV7220-2S, SV7220G2-N, SV7220G2-P, SV7220G2-S, SV7220G2-V, NM10GR, and NM10GS; any other activities, products and/or services involving the products identified above; and any other activities, products and/or services that practice and/or support similarly infringing LSO functionality (collectively, "the '104 Accused Products"). The '104 Accused Products are non-limiting examples that were identified based on publicly available information, and Alacritech reserves the right to identify additional infringing activities, products and services, including, for example, on the basis of information obtained during discovery.

119. As just one non-limiting example, set forth below (with claim language in italics) is a description of Defendants' infringement of exemplary claim 1 of the '104 patent in connection with the Wiwynn SV7220G2 Server configured with a Wiwynn NM10GR Mezzanine Card. This description is based on publicly available information. Alacritech reserves the right to modify this description, including, for example, on the basis of information about the '104 Accused Products that it obtains during discovery.

1(a) *A method for communication involving a computer, a network, and a network interface device of the computer, the network interface device being coupled to the network, the method comprising: receiving, by the network interface device from the computer, a command to transmit application data from the computer to the network*; – Defendants make, use, sell, offer for sale, and/or import servers, network interface devices (e.g., controllers and cards), and other network products that support infringing LSO functionality. One example of an infringing products is the Wiwynn SV7220G2 Server (the "SV7220G2 Server") configured with a Wiwynn NM10GR Mezzanine Card (the "NM10GR Card") comprising an Intel X550 controller (the

"X550 Controller"). *See*, *e.g.*, "Wiwynn 2016 OCP US Summit Product Brochure," available at https://www.circleb.eu/wp-content/uploads/2016/04/OCP-US-Summit-OPS.pdf ("OCP Product Brochure"); "Wiwynn SV7220G2 Series Datasheet," available at http://www.wiwynn.com/usr_files/20150413140223_Datasheet%20-%20SV7220G2_150413.pdf ("SV7220G2 Datasheet"). To the extent the preamble is limiting, the SV7220G2 Server is a computer with a network interface device comprising the NM10GR Card that is coupled to a network. In connection with LSO functionality, the network interface device of the SV7220G2 Server receives a command from the computer to send application data from the computer to the network (e.g., for transmission to another server). The "Intel Ethernet Controller X550 Datasheet," Revision 2.1 (May 2016), available at http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/ethernet-x550-datasheet.pdf (the "X550 Datasheet") provides an overview of the X550 Controller and explains how it interacts with the server, including, for example, describing in Section 7.2.4 (excerpts of which appear below) how it receives a command from the computer to send application data from the computer to the network.



7.2.4 TCP and UDP Segmentation

Hardware TCP segmentation is one of the offloading options supported by the Windows* and Linux* TCP/IP stack. This is often referred to as Large Send offloading or TSO. This feature enables the TCP/IP stack to pass to the network device driver a message to be transmitted that is bigger than the Maximum Transmission Unit (MTU) of the medium. It is then the responsibility of the device driver and hardware to divide the TCP message into MTU size frames that have appropriate layer 2 (Ethernet), 3 (IP), and 4 (TCP) headers. These headers must include sequence number, checksum fields, options and flag values as required. Note that some of these values (such as the checksum values) are unique for each packet of the TCP message, and other fields such as the source IP Address is constant for all packets associated with the TCP message.

***

7.2.4.2 Transmission Process

The transmission process involves the following:

- The protocol stack receives from an application a block of data that is to be transmitted.
- The protocol stack calculates the number of packets required to transmit this block based on the MTU size of the media and required packet headers.

- The stack interfaces with the device driver and passes the block down with the appropriate header information: Ethernet, IP and TCP / UDP headers.
- The stack interfaces with the device driver and commands the driver to send the individual packet. The device driver sets up the interface to the hardware (via descriptors) for the TCP / UDP segmentation.
- The hardware transfers (DMA's) the packet data and performs the Ethernet packet segmentation and transmission based on offset and payload length parameters in the TCP/IP or UDP/IP context descriptor including:
  - — Packet encapsulation
  - — Header generation and field updates including IPv4/IPv6 and TCP/UDP checksum generation.
- The driver returns ownership of the block of data to the NOS when the hardware has completed the DMA transfer of the entire data block.

***

7.2.4.5 TCP and UDP Segmentation Indication

Software indicates a TCP/UDP segmentation transmission context to the hardware by setting up a TCP/IP or UDP/IP context transmit descriptor (see Section 7.2.3). The purpose of this descriptor is to provide information to the hardware to be used during the TCP / UDP segmentation offload process.

Setting the *TSE* bit in the *DCMD* field to one (in the data descriptor) indicates that this descriptor refers to the segmentation context (as opposed to the normal checksum offloading context). This causes the checksum offloading, packet length, header length, and maximum segment size parameters to be loaded from the descriptor into the device.

The TCP/UDP segmentation prototype header is taken from the packet data itself. Software must identity the type of packet that is being sent (IPv4/IPv6, TCP/UDP, other), calculate appropriate checksum off loading values for the desired checksums, and then calculate the length of the header that is pre-appended. The header can be up to 240 bytes in length.

Once the TCP/UDP segmentation context has been set, the next descriptor provides the initial data to transfer. This first descriptor(s) must point to a packet of the type indicated. Furthermore, the data it points to might need to be modified by software as it serves as the prototype header for all packets within the TC /UDP segmentation context. The following sections describe the supported packet types and the various updates that are performed by hardware. This should be used as a guide to determine what must be modified in the original packet header to make it a suitable prototype header.

1(b) *sending, by the network interface device to the network, data corresponding to the command, including prepending a transport layer header to at least some of the data*; – In connection with the LSO functionality discussed above, the network interface device of the

SV7220G2 Server incorporating the NM10GR Card sends the data corresponding to the command to the network and, as a part of sending the data, it prepares and prepends a transport layer header (e.g., a TCP header) to at least some of the data before it is sent. The X550 Datasheet describes this operation in the sections shown above and, also, for example, in Sections 7.2.4.4 and 7.2.4.7.

7.2.4.4 Packet Format

A TCP / UDP message can be as large as 256 KB and is generally fragmented across multiple pages in host memory. The X550 partitions the data packet into standard Ethernet frames prior to transmission. The X550 supports calculating the Ethernet, IP, TCP, and UDP headers, including checksum, on a frame-by-frame basis. For tunneled packets (NVGRE, VXLAN), The X550 also supports updating the outer IPv4 header.

Table 7-41 TCP/IP and UDP/IP Packet Format Sent by Host

Pseudo Header				Data
Ethernet	Optional tunnel header	IPv4/IPv6	TCP/UDP	DATA (full TCP/UDP message)

Table 7-42 Packets Format Sent by Device

Pseudo Header (updated)	Data (first MSS)	FCS	...	Pseudo Header (updated)	Data (Next MSS)	FCS	...

***

7.2.4.7 IP/TCP/UDP Header Updating

IP/TCP and IP/UDP header is updated for each outgoing frame based on the header prototype that hardware DMA's from the first descriptor(s). The checksum fields and other header information are later updated on a frame-by-frame basis. The updating process is performed concurrently with the packet data fetch.

The following sections define what fields are modified by hardware during the TCP/UDP segmentation process by the X550.

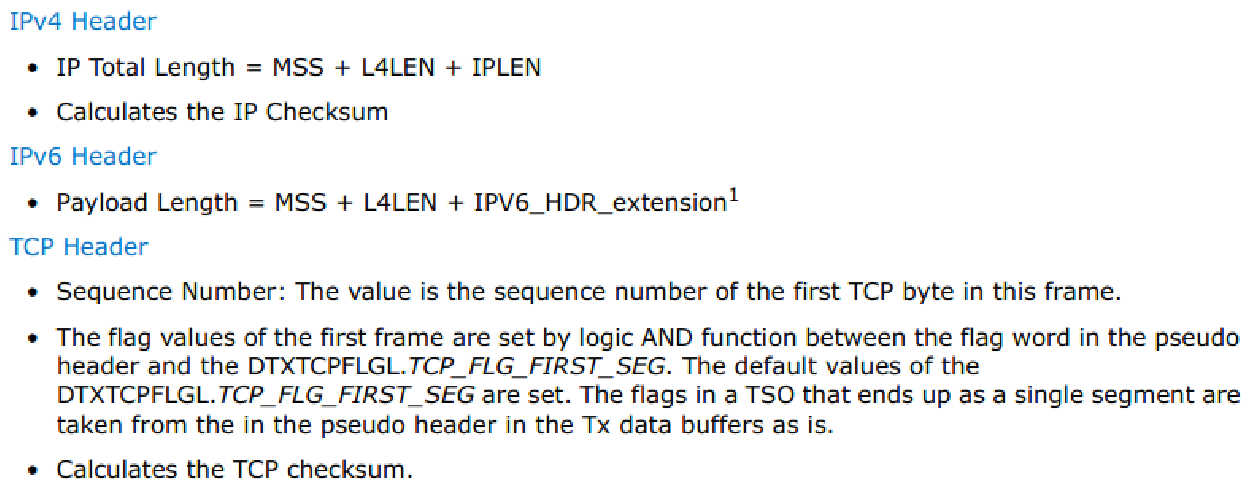
7.2.4.7.1 TCP/IP/UDP Header for the First Frame

The hardware makes the following changes to the headers of the first packet that is derived from each TCP segmentation context.

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follows:

- IP Total Length = OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + MSS
- Calculates the Outer IP Checksum



IPv4 Header

- IP Total Length = MSS + L4LEN + IPLEN
- Calculates the IP Checksum

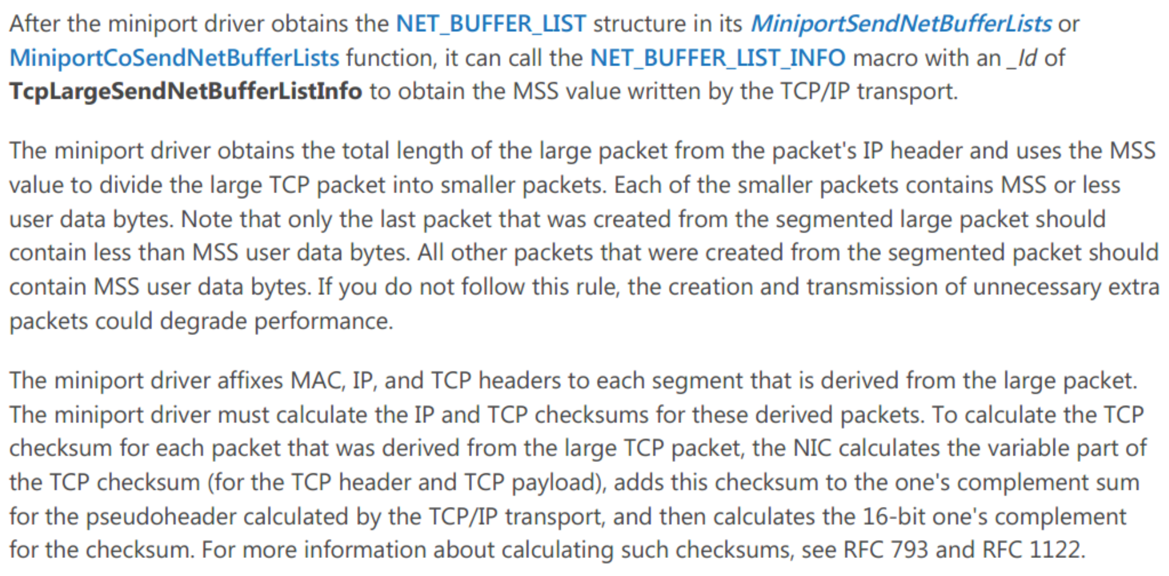
IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension[1]

TCP Header

- Sequence Number: The value is the sequence number of the first TCP byte in this frame.
- The flag values of the first frame are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGL.*TCP_FLG_FIRST_SEG*. The default values of the DTXTCPFLGL.*TCP_FLG_FIRST_SEG* are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculates the TCP checksum.

Microsoft also describes this interaction between a network interface device supporting infringing LSO functionality and a server (such as the SV7220G2 Server incorporating the NM10GR Card) running the Microsoft Windows Server Operating System, which enables LSO by default, including the operations described below when sending data in connection with LSO. *See, e.g.*, "Offloading the Segmentation of Large TCP Packets," available at https://msdn.microsoft.com/en-us/library/windows/hardware/ff568840.aspx.

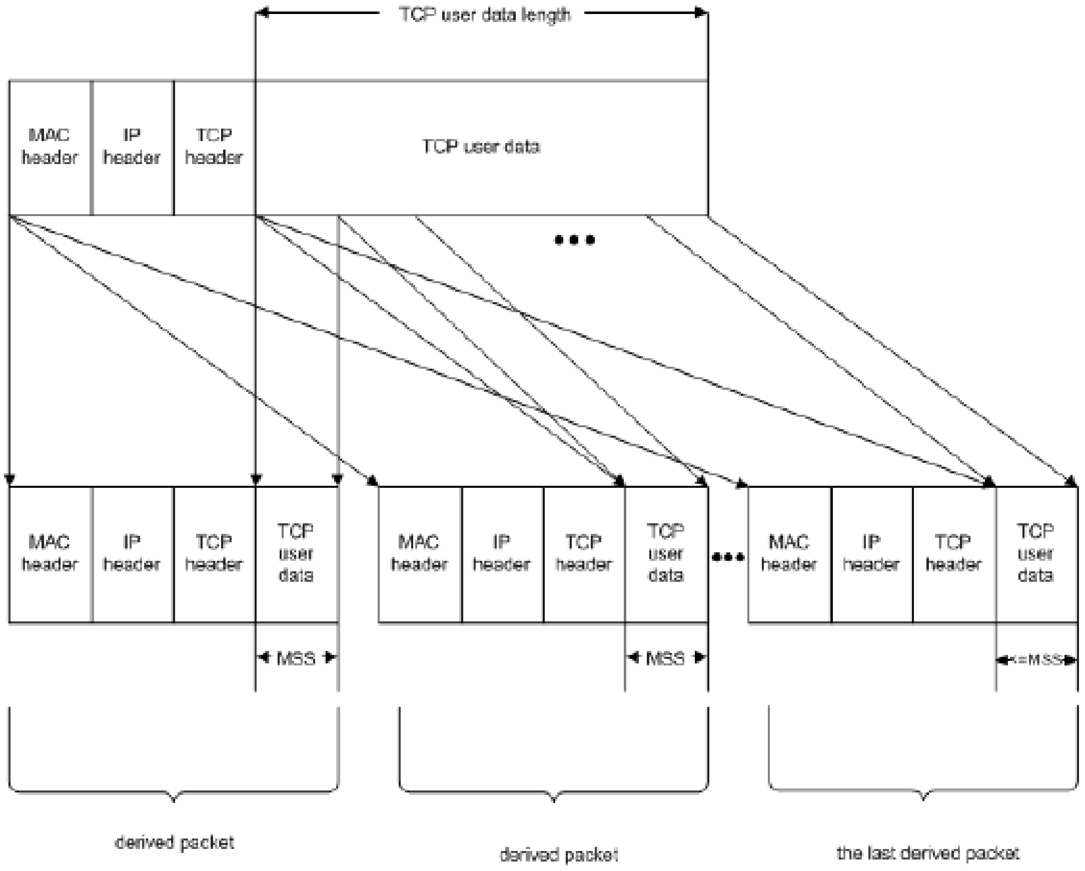


After the miniport driver obtains the NET_BUFFER_LIST structure in its *MiniportSendNetBufferLists* or MiniportCoSendNetBufferLists function, it can call the NET_BUFFER_LIST_INFO macro with an *_Id* of **TcpLargeSendNetBufferListInfo** to obtain the MSS value written by the TCP/IP transport.

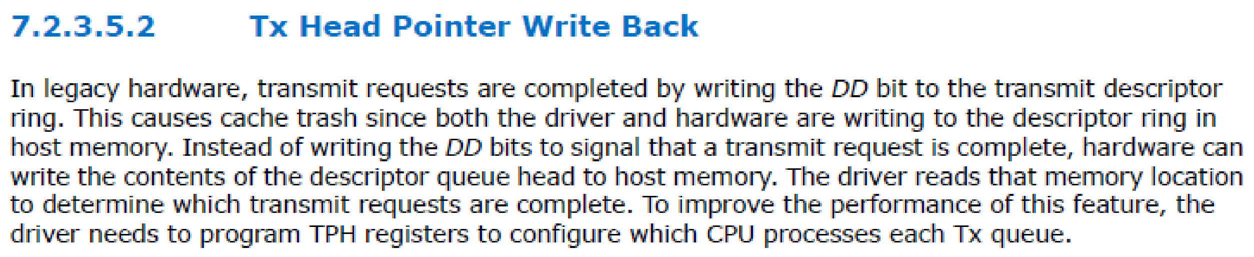
The miniport driver obtains the total length of the large packet from the packet's IP header and uses the MSS value to divide the large TCP packet into smaller packets. Each of the smaller packets contains MSS or less user data bytes. Note that only the last packet that was created from the segmented large packet should contain less than MSS user data bytes. All other packets that were created from the segmented packet should contain MSS user data bytes. If you do not follow this rule, the creation and transmission of unnecessary extra packets could degrade performance.

The miniport driver affixes MAC, IP, and TCP headers to each segment that is derived from the large packet. The miniport driver must calculate the IP and TCP checksums for these derived packets. To calculate the TCP checksum for each packet that was derived from the large TCP packet, the NIC calculates the variable part of the TCP checksum (for the TCP header and TCP payload), adds this checksum to the one's complement sum for the pseudoheader calculated by the TCP/IP transport, and then calculates the 16-bit one's complement for the checksum. For more information about calculating such checksums, see RFC 793 and RFC 1122.

The following figure shows the segmentation of a large packet.



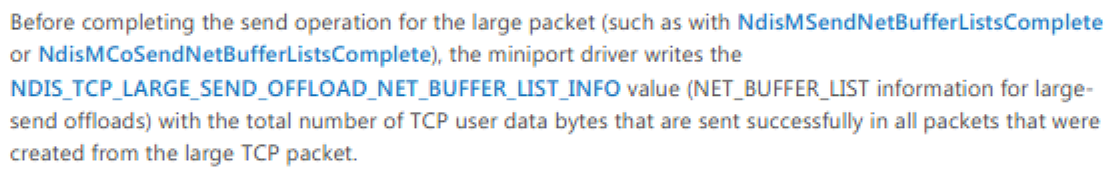
1(c) *sending, by the network interface device to the computer, a response to the command indicating that the data has been sent from the network interface device to the network, prior to receiving, by the network interface device from the network, an acknowledgement (ACK) that all the data corresponding to the command has been received; and* – In connection with the LSO functionality discussed above, the network interface device of the SV7220G2 Server incorporating the NM10GR Card sends the computer a response to the command indicating that the data has been sent to the network, and it sends the response before it receives an ACK from the network that all data corresponding to the command has been received. This is described, for example, in Section 7.2.3.5.2 of the X550 Datasheet.



7.2.3.5.2 Tx Head Pointer Write Back

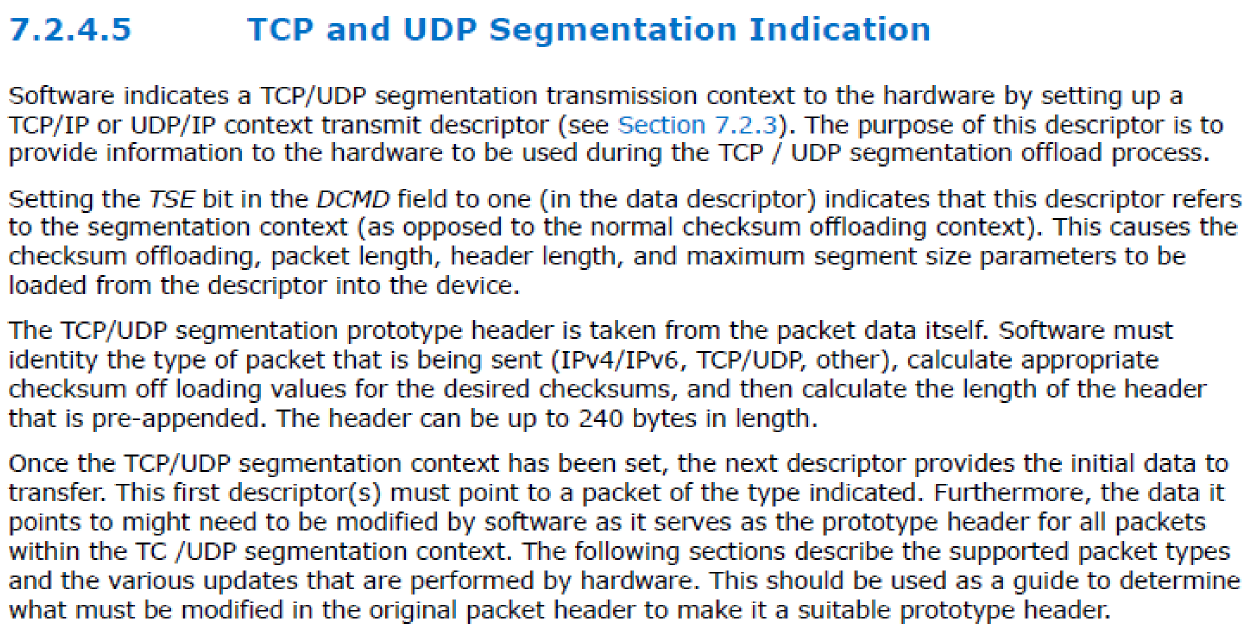
In legacy hardware, transmit requests are completed by writing the *DD* bit to the transmit descriptor ring. This causes cache trash since both the driver and hardware are writing to the descriptor ring in host memory. Instead of writing the *DD* bits to signal that a transmit request is complete, hardware can write the contents of the descriptor queue head to host memory. The driver reads that memory location to determine which transmit requests are complete. To improve the performance of this feature, the driver needs to program TPH registers to configure which CPU processes each Tx queue.

Microsoft also describes this operation. *See, e.g.*, "Offloading the Segmentation of Large TCP Packets," available at https://msdn.microsoft.com/en-us/library/windows/hardware/ff568840.aspx.



Before completing the send operation for the large packet (such as with NdisMSendNetBufferListsComplete or NdisMCoSendNetBufferListsComplete), the miniport driver writes the NDIS_TCP_LARGE_SEND_OFFLOAD_NET_BUFFER_LIST_INFO value (NET_BUFFER_LIST information for large-send offloads) with the total number of TCP user data bytes that are sent successfully in all packets that were created from the large TCP packet.

1(d) *maintaining, by the network interface device, a Transport Control Protocol (TCP) connection that the command, the data and the ACK correspond to.* – In connection with performing LSO functionality discussed above and transmitting data, the network interface device of the SV7220G2 Server incorporating the NM10GR Card maintains a TCP connection based on information provided by the computer, to which the command, the data being sent, and the ACK correspond. This is described in the materials discussed above, including, for example, in Section 7.2.4.5 of the X550 Datasheet.



7.2.4.5 TCP and UDP Segmentation Indication

Software indicates a TCP/UDP segmentation transmission context to the hardware by setting up a TCP/IP or UDP/IP context transmit descriptor (see Section 7.2.3). The purpose of this descriptor is to provide information to the hardware to be used during the TCP / UDP segmentation offload process.

Setting the *TSE* bit in the *DCMD* field to one (in the data descriptor) indicates that this descriptor refers to the segmentation context (as opposed to the normal checksum offloading context). This causes the checksum offloading, packet length, header length, and maximum segment size parameters to be loaded from the descriptor into the device.

The TCP/UDP segmentation prototype header is taken from the packet data itself. Software must identity the type of packet that is being sent (IPv4/IPv6, TCP/UDP, other), calculate appropriate checksum off loading values for the desired checksums, and then calculate the length of the header that is pre-appended. The header can be up to 240 bytes in length.

Once the TCP/UDP segmentation context has been set, the next descriptor provides the initial data to transfer. This first descriptor(s) must point to a packet of the type indicated. Furthermore, the data it points to might need to be modified by software as it serves as the prototype header for all packets within the TC /UDP segmentation context. The following sections describe the supported packet types and the various updates that are performed by hardware. This should be used as a guide to determine what must be modified in the original packet header to make it a suitable prototype header.

120. As of no later than the filing and service of this Complaint, Defendants are also indirectly infringing the '104 patent.

121. Defendants have actual knowledge of Alacritech's rights in the '104 patent and details of Defendants' infringement of the '104 patent based on at least the filing and service of this Complaint.

122. Defendants make, use, import, offer for sale, and/or sell the '104 Accused Products (including, for example, the products described above) with knowledge of or willful blindness to the fact that their actions will induce their customers to infringe the '104 patent. Defendants induce others to infringe the '104 patent in violation of 35 U.S.C. § 271 by encouraging and facilitating others to practice the '104 patent's inventions for accelerated network communications with intent that those performing the acts infringe the '104 patent. For example, Defendants sell and/or provide '104 Accused Products that perform infringing LSO functionality by default. When Defendants' customers use the '104 Accused Products for their intended purpose, they practice one or more claims of the '104 patent. Defendants instructs their customers how to set up and use '104 Accused Products. In some cases, Defendants even sets up

'104 Accused Products for their customers. Once they have been set up, Defendants' customers infringe one or more claims of the '104 patent simply by using the '104 Accused Products. By selling and/or providing their customers '104 Accused Products that infringe by default when used for their intended purpose, and by setting up for their customers and/or instructing their customers how to set up and use those products, Defendants induce those customers to infringe the '104 patent.

123. Defendants also contribute to the infringement of the '104 patent in violation of 35 U.S.C. § 271. Defendants know that infringing components of the '104 Accused Products are especially made or especially adapted for use in the infringement of the '104 patent. The infringing components of these products are not staple articles or commodities of commerce suitable for substantial non-infringing use, and the infringing components of these products are a material part of the invention of the '104 patent. The '104 Accused Products contain infringing components such as specialized hardware and/or software for performing infringing LSO functionality, which are separable from the '104 Accused Products, material to practicing the '104 patent's inventions for accelerated network communications, and have no substantial non-infringing use. Accordingly, Defendants are also contributing to the direct infringement of the '104 patent by their customers.

124. Defendants are not licensed or otherwise authorized by Alacritech to practice, contributorily practice and/or induce third parties to practice the claims of the '104 patent.

125. By reason of Defendants' infringing activities, Alacritech has suffered, and will continue to suffer, substantial damages in an amount to be proven at trial. Alacritech is entitled to a money judgment in an amount adequate to compensate for Defendants' infringement, but in

no event less than a reasonable royalty for the use made of the invention by Defendants, together with interest and costs as fixed by the Court.

126. Defendants' continuing acts of infringement have caused and will continue to cause Alacritech irreparable harm, for which Alacritech has no adequate remedy at law, unless Defendants' continuing acts of infringement are enjoined by the Court. The hardships that an injunction would impose are less than those faced by Alacritech should an injunction not issue. The public interest would be served by issuance of an injunction.

127. Defendants' infringement of the '104 patent is exceptional and entitles Alacritech to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

PRAYER FOR RELIEF

Alacritech respectfully requests the following relief from this Court:

A. A judgment that Defendants have infringed each and every one of the Asserted Patents;

B. A preliminary and permanent injunction against Defendants, their respective officers, agents, servants, employees, attorneys, parent and subsidiary corporations, assigns and successors in interest, and those persons in active concert or participation with them, enjoining them from infringement, inducement of infringement, and contributory infringement of each and every one of the Asserted Patents, including but not limited to an injunction against making, using, selling, and/or offering for sale within the United States, and/or importing into the United States, any products and/or services that infringe the Asserted Patents;

C. Damages adequate to compensate Alacritech for Defendants' infringement of the Asserted Patents pursuant to 35 U.S.C. § 284;

D. Prejudgment interest;

E. Post-judgment interest;

F. A judgment and order finding that this is an exceptional case within the meaning of 35 U.S.C. § 285, and an award to Alacritech of its attorneys' fees, costs and expenses incurred in connection with this Action; and

G. Such other relief as the Court deems just and equitable.

DEMAND FOR JURY TRIAL

Pursuant to Rule 38 of the Federal Rules of Civil Procedure, Alacritech demands a trial by jury on all matters and issues triable by jury.

Dated: June 30, 2016

QUINN EMANUEL URQUHART &
SULLIVAN, LLP

/s/ Claude M. Stern by permission Claire A. Henry
Claude M. Stern
California State Bar No. 96737
claudestern@quinnemanuel.com
QUINN EMANUEL URQUHART &
SULLIVAN, LLP
555 Twin Dolphin Drive, 5th Floor
Redwood Shores, CA 94065
Telephone: (650) 801-5000
Facsimile: (650) 801-5100

Joseph M. Paunovich
joepaunovich@quinnemanuel.com
California State Bar No. 228222
Jordan Brock Kaericher
California State Bar No. 265953
jordankaericher@quinnemanuel.com
QUINN EMANUEL URQUHART &
SULLIVAN, LLP
865 South Figueroa Street, 10th Floor
Los Angeles, CA 90017
Telephone: (213) 443-3000
Facsimile: (213) 443-3100

T. John Ward, Jr.
Texas State Bar No. 00794818
jw@wsfirm.com
Claire Abernathy Henry
Texas State Bar No. 24053063

claire@wsfirm.com
Andrea L. Fair
Texas State Bar No. 24078488
WARD, SMITH & HILL, PLLC
1507 Bill Owens Parkway
Longview, Texas 75604
Telephone: (903) 757-6400
Facsimile: (903) 757-2323

ATTORNEYS FOR PLAINTIFF
ALACRITECH, INC.