US009270602B1

(12) **United States Patent**
Mimms et al.

(10) **Patent No.: US 9,270,602 B1**
(45) **Date of Patent: Feb. 23, 2016**

(54) **TRANSMIT RATE PACING OF LARGE NETWORK TRAFFIC BURSTS TO REDUCE JITTER, BUFFER OVERRUN, WASTED BANDWIDTH, AND RETRANSMISSIONS**

(71) Applicant: **F5 Networks, Inc.**, Seattle, WA (US)

(72) Inventors: **Alan B. Mimms**, Spokane, CA (US); **Timothy S. Michels**, Greenacres, CA (US); **Jonathan M. Hawthorne**, Seattle, WA (US); **William R. Baumann**, Seattle, WA (US)

(73) Assignee: **F5 Networks, Inc.**, Seattle, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 258 days.

(21) Appl. No.: **13/732,337**

(22) Filed: **Dec. 31, 2012**

(51) **Int. Cl.**

H04L 12/819	(2013.01)
H04L 12/879	(2013.01)
G06F 13/28	(2006.01)

(52) **U.S. Cl.**
CPC ***H04L 47/21*** (2013.01); *G06F 13/28* (2013.01); *H04L 49/901* (2013.01)

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,914,650	A *	4/1990	Sriram	370/235
5,388,237	A	2/1995	Sodos	
5,477,541	A *	12/1995	White et al.	370/392
5,699,361	A *	12/1997	Ding et al.	370/431
5,761,534	A	6/1998	Lundberg et al.	
5,828,835	A	10/1998	Isfeld et al.	
5,941,988	A	8/1999	Bhagwat et al.	
6,026,090	A *	2/2000	Benson et al.	370/395.7
6,026,443	A	2/2000	Oskouy et al.	
6,070,219	A *	5/2000	McAlpine et al.	710/263
6,115,802	A	9/2000	Tock et al.	
6,347,337	B1 *	2/2002	Shah et al.	709/224
6,388,989	B1 *	5/2002	Malhotra	370/229
6,529,508	B1	3/2003	Li et al.	
6,574,220	B1 *	6/2003	Petty	370/395.4
6,700,871	B1	3/2004	Harper et al.	
6,748,457	B2	6/2004	Fallon et al.	
6,781,990	B1	8/2004	Puri et al.	
6,785,236	B1 *	8/2004	Lo et al.	370/235
6,820,133	B1	11/2004	Grove et al.	
6,934,776	B2 *	8/2005	Connor et al.	710/60

(Continued)

FOREIGN PATENT DOCUMENTS

EP	1813084	A1	8/2007
WO	WO 2006/055494	A1	5/2006

OTHER PUBLICATIONS

Cavium Networks, "Cavium Networks Product Selector Guide—Single & Multi-Core MIPS Processors, Security Processors and Accelerator Boards," 2008, pp. 1-44, Mountain View, CA, US.

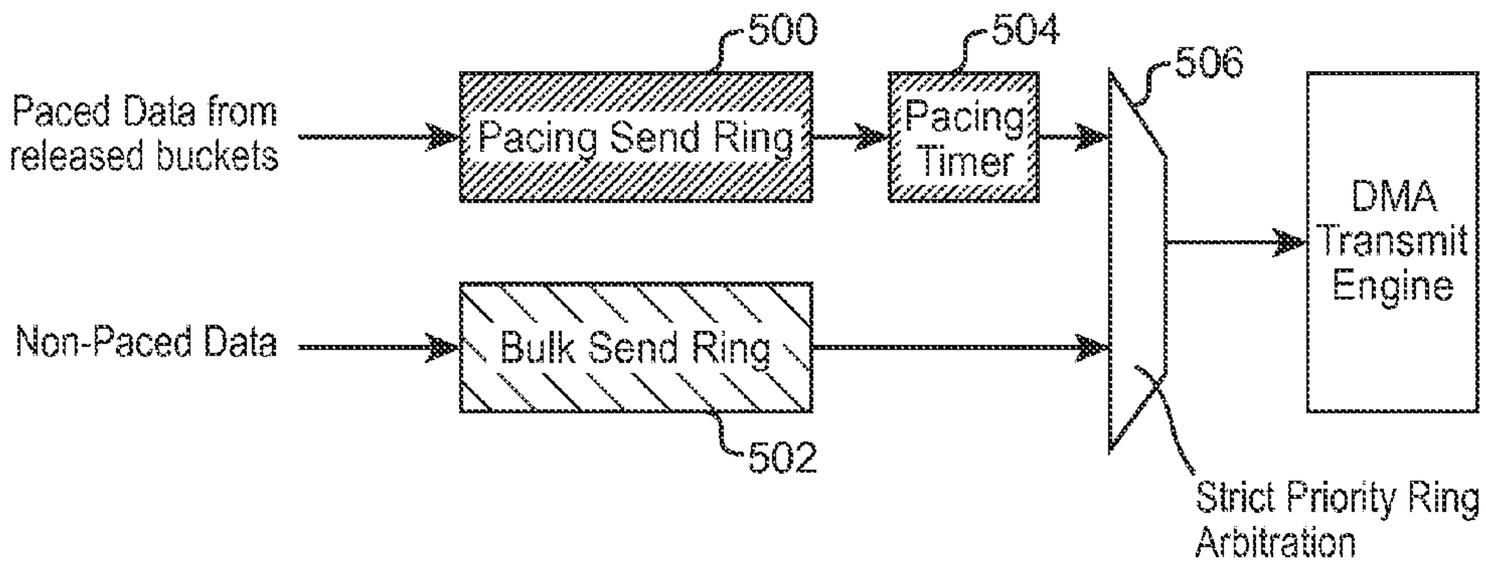
(Continued)

Primary Examiner — Dung B Huynh
(74) *Attorney, Agent, or Firm* — LeClairRyan, a Professional Corporation

(57) **ABSTRACT**

A system, method and medium is disclosed which includes selecting, at a software component of a network traffic management device, a first bucket having a first predetermined transmit time. The disclosure includes populating one or more selected data packet descriptors associated with one or more corresponding data packets in the first bucket. The disclosure includes releasing the first bucket to a hardware component of the network traffic management device, wherein the hardware component processes the one or more data packet descriptors of the first bucket for the first predetermined transmit time.

18 Claims, 8 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

7,046,628	B2 *	5/2006	Luhmann et al.	370/230
7,065,630	B1	6/2006	Ledebohm et al.	
7,107,348	B2	9/2006	Shimada et al.	
7,124,196	B2 *	10/2006	Hooper	709/232
7,142,540	B2	11/2006	Hendel et al.	
7,164,678	B2 *	1/2007	Connor	370/392
7,236,491	B2	6/2007	Tsao et al.	
7,281,030	B1	10/2007	Davis	
7,324,525	B2	1/2008	Fuhs et al.	
7,327,674	B2 *	2/2008	Eberle et al.	370/230
7,349,405	B2 *	3/2008	Deforche	370/395.4
7,355,977	B1	4/2008	Li	
7,376,772	B2	5/2008	Fallon	
7,403,542	B1	7/2008	Thompson	
7,420,931	B2	9/2008	Nanda et al.	
7,475,122	B2	1/2009	Azpitarte	
7,478,186	B1	1/2009	Onufryk et al.	
7,496,695	B2	2/2009	Go et al.	
7,500,028	B2	3/2009	Yamagishi	
7,512,721	B1	3/2009	Olson	
7,533,197	B2	5/2009	Leonard et al.	
7,558,910	B2	7/2009	Alverson et al.	
7,571,299	B2	8/2009	Loeb	
7,647,416	B2	1/2010	Chiang et al.	
7,649,882	B2 *	1/2010	Stiliadis	370/390
7,657,659	B1	2/2010	Lambeth et al.	
7,668,727	B2	2/2010	Mitchell et al.	
7,668,851	B2	2/2010	Triplett	
7,729,239	B1	6/2010	Aronov et al.	
7,734,809	B2	6/2010	Joshi et al.	
7,735,099	B1	6/2010	Micalizzi, Jr.	
7,742,412	B1	6/2010	Medina	
7,784,093	B2	8/2010	Deng et al.	
7,826,487	B1	11/2010	Mukerji et al.	
7,877,524	B1	1/2011	Annem et al.	
7,916,728	B1	3/2011	Mimms	
7,929,433	B2 *	4/2011	Husak et al.	370/229
7,975,025	B1	7/2011	Szabo et al.	
8,006,016	B2	8/2011	Muller et al.	
8,103,809	B1	1/2012	Michels et al.	
8,112,491	B1	2/2012	Michels et al.	
8,112,594	B2	2/2012	Giacomoni et al.	
8,279,865	B2	10/2012	Giacomoni et al.	
8,306,036	B1	11/2012	Bollay et al.	
8,346,993	B2	1/2013	Michels et al.	
8,447,884	B1	5/2013	Baumann	
8,880,632	B1	11/2014	Michels et al.	
8,880,696	B1	11/2014	Michels et al.	
2001/0038629	A1 *	11/2001	Shinohara	370/394
2002/0156927	A1	10/2002	Boucher et al.	
2003/0067930	A1	4/2003	Salapura et al.	
2003/0204636	A1	10/2003	Greenblat et al.	
2004/0032830	A1 *	2/2004	Bly et al.	370/235
2004/0062245	A1 *	4/2004	Sharp et al.	370/392
2004/0202161	A1	10/2004	Stachura et al.	
2004/0249881	A1	12/2004	Jha et al.	
2004/0249948	A1	12/2004	Sethi et al.	
2004/0267897	A1	12/2004	Hill et al.	
2005/0007991	A1	1/2005	Ton et al.	
2005/0022623	A1	2/2005	Reiche et al.	
2005/0083952	A1	4/2005	Swain	
2005/0091390	A1 *	4/2005	Helmer et al.	709/230
2005/0114559	A1	5/2005	Miller	
2005/0141427	A1 *	6/2005	Bartky	370/235
2005/0175014	A1	8/2005	Patrick	
2005/0213570	A1	9/2005	Stacy et al.	
2005/0226234	A1 *	10/2005	Sano et al.	370/378
2006/0007928	A1	1/2006	Sangillo	
2006/0104303	A1	5/2006	Makineni et al.	
2006/0221832	A1	10/2006	Muller et al.	
2006/0221835	A1	10/2006	Sweeney	
2006/0224820	A1 *	10/2006	Cho et al.	711/103
2006/0235996	A1	10/2006	Wolde et al.	
2006/0288128	A1	12/2006	Moskalev et al.	
2007/0162619	A1 *	7/2007	Aloni et al.	709/250
2008/0126509	A1	5/2008	Subramanian et al.	
2008/0184248	A1	7/2008	Barua et al.	
2008/0201772	A1 *	8/2008	Mondaeev et al.	726/13
2008/0219279	A1	9/2008	Chew	
2009/0003204	A1	1/2009	Okholm et al.	
2009/0016217	A1	1/2009	Kashyap	
2009/0089619	A1	4/2009	Huang et al.	
2009/0154459	A1 *	6/2009	Husak et al.	370/390
2009/0222598	A1	9/2009	Hayden	
2009/0248911	A1	10/2009	Conroy et al.	
2009/0279559	A1 *	11/2009	Wong et al.	370/412
2010/0082849	A1	4/2010	Millet et al.	
2010/0085875	A1 *	4/2010	Solomon et al.	370/235
2010/0094945	A1	4/2010	Chan et al.	
2011/0228781	A1 *	9/2011	Izenberg et al.	370/392
2013/0250777	A1 *	9/2013	Ziegler	370/248

OTHER PUBLICATIONS

"Chapter 15, Memory Mapping and DMA," Memory Management in Linux, ch15.13676, accessed on Jan. 25, 2005, pp. 412-463.

Comtech AHA Corporation, "Comtech AHA Announces 3.0 Gbps GZIP Compression/Decompression Accelerator AHA362-PCIX offers high-speed GZIP compression and decompression," www.aha.com, Apr. 20, 2005, pp. 1-2, Moscow, ID, USA.

Comtech AHA Corporation, "Comtech AHA Announces GZIP Compression and Decompression IC Offers the highest speed and compression ratio performance in hardware on the market," www.aha.com, Jun. 26, 2007, pp. 1-2, Moscow, ID, USA.

EventHelix, "DMA and Interrupt Handling," <http://www.eventhelix.com/RealtimeMantra/FaultHandling/dma_interrupt_handling.htm>, Jan. 29, 2010, pp. 1-4, EventHelix.com.

Alteon Websystems Inc., "Gigabit Ethernet/PCI Network Interface Card; Host/NIC Software Interface Definition," Jul. 1999, pp. 1-80, Revision 12.4.13, P/N 020001, San Jose, California.

Harvey et al., "DMA Fundamentals on Various PC Platforms," Application Note 011, Apr. 1999, pp. 1-20, National Instruments Corporation.

Mangino, John, "Using DMA with High Performance Peripherals to Maximize System Performance," WW TMS470 Catalog Applications, SPNA105 Jan. 2007, pp. 1-23.

Mogul, Jeffrey C., "The Case for Persistent-Connection HTTP," SIGCOMM '95, Digital Equipment Corporation Western Research Laboratory, 1995, pp. 1-15, Cambridge, Maine.

Cavium Networks, "NITROX™ XL Security Acceleration Modules PCI 3V or 3V/5V-Universal Boards for SSL and IPSec," at http://www.Caviumnetworks.com, 2002, pp. 1, Mountain View, CA USA.

Cavium Networks, "PCI, PCI-X," at (http://www.cavium.com/acceleration_boards_PCI_PCI-X.htm (Downloaded Oct. 2008), Cavium Networks—Products > Acceleration Boards > PCI, PCI-X).

"Plan 9 kernel history: overview / file list / diff list," <http://switch.com/cgi-bin/plan9history.cgi?f=2001/0126/pc/etherga620.com>, accessed Oct. 22, 2007, pp. 1-16.

Rabinovich et al., "DHTTP: An Efficient and Cache-Friendly Transfer Protocol for the Web," IEEE/ACM Transactions on Networking, Dec. 2004, pp. 1007-1020, vol. 12, No. 6.

Salchow, Jr., KJ, "Clustered Multiprocessing: Changing the Rules of the Performance Game," F5 White Paper, Jan. 2008, pp. 1-11, F5 Networks, Inc.

Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," Network Working Group, RFC 2001, Jan. 1997, pp. 1-6.

EventHelix, "TCP—Transmission Control Protocol (TCP Fast Retransmit and Recovery)," Mar. 28, 2002, pp. 1-5, EventHelix.com.

Wadge, Wallace, "Achieving Gigabit Performance on Programmable Ethernet Network Interface Cards," May 29, 2001, pp. 1-9.

Welch, Von, "A User's Guide to TCP Windows," http://www.vonwelch.com/report/tcp_windows, updated 1996, last accessed Jan. 29, 2010, pp. 1-5.

Wikipedia, "Direct memory access," <http://en.wikipedia.org/wiki/Direct_memory_access>, accessed Jan. 29, 2010, pp. 1-6.

* cited by examiner

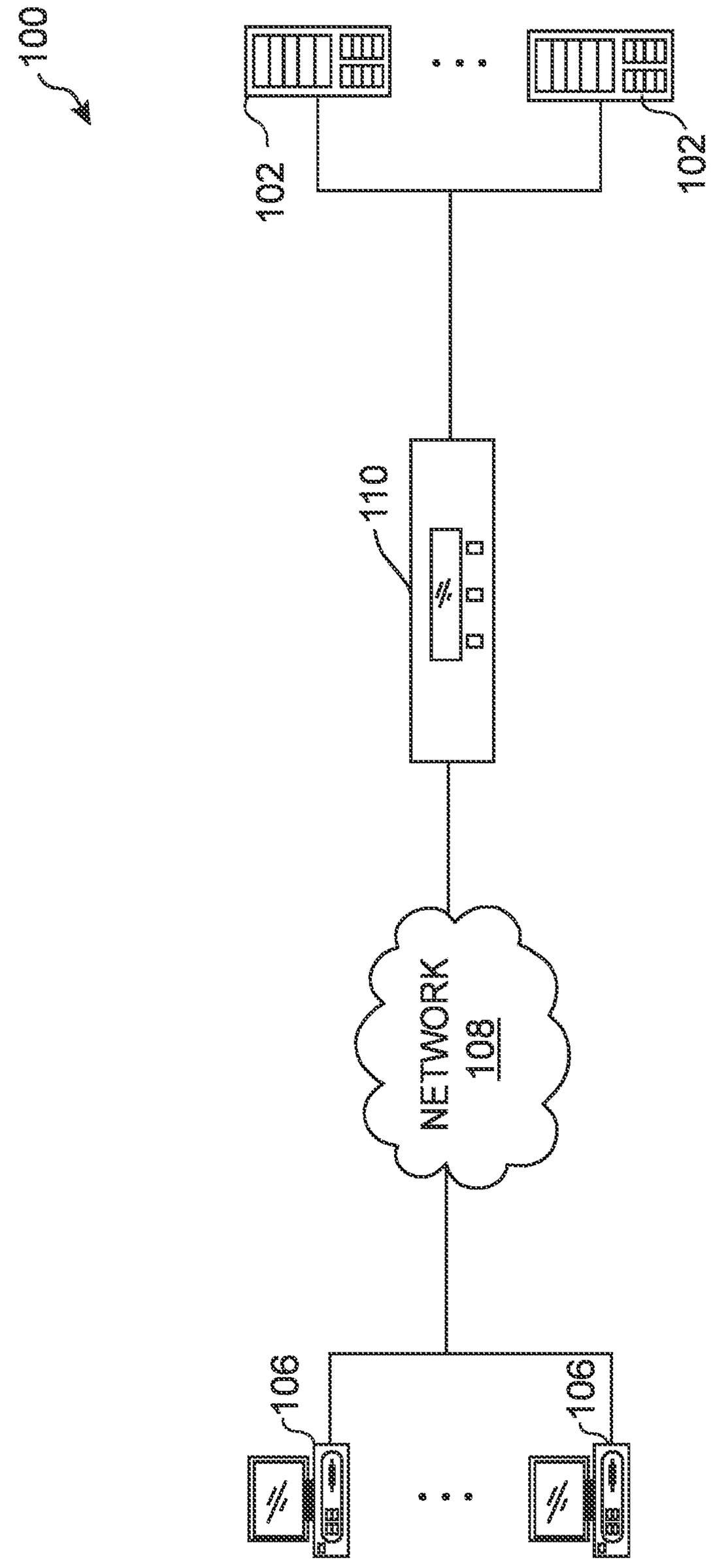


FIG. 1

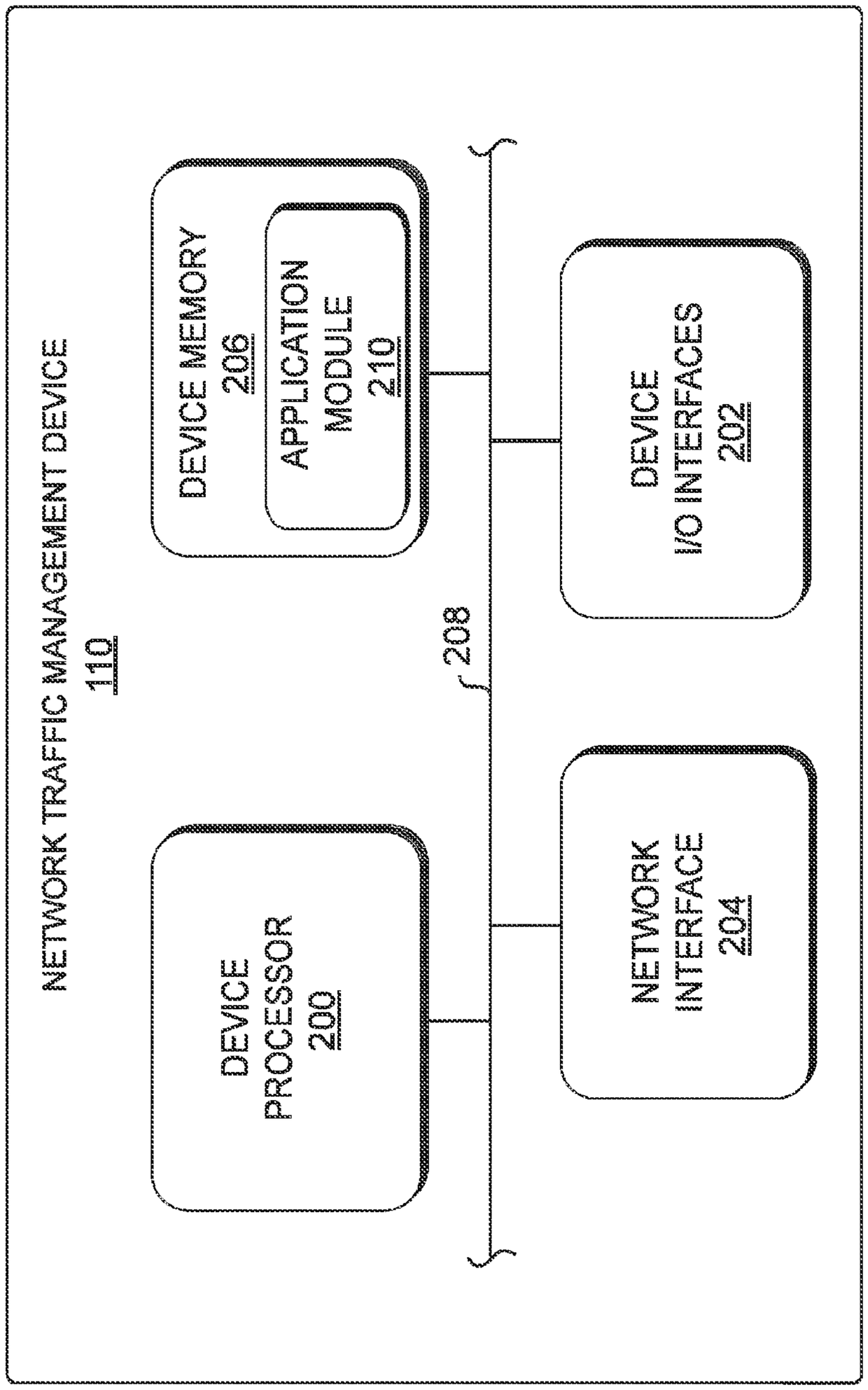

NETWORK TRAFFIC MANAGEMENT DEVICE
110
DEVICE PROCESSOR 200
DEVICE MEMORY 206
APPLICATION MODULE 210
208
NETWORK INTERFACE 204
DEVICE I/O INTERFACES 202

FIG. 2A

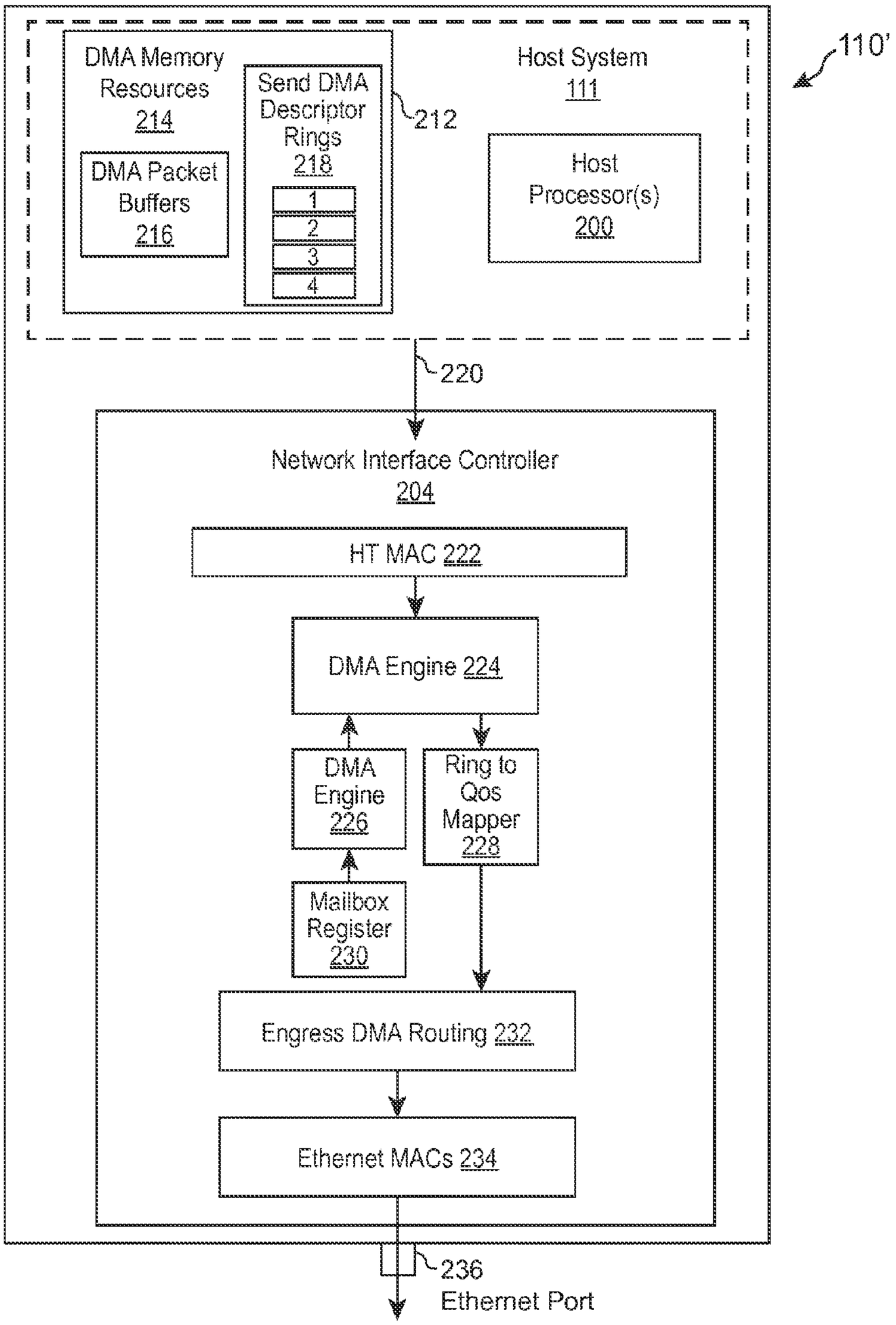

110'
DMA Memory Resources 214
DMA Packet Buffers 216
Send DMA Descriptor Rings 218
1
2
3
4
212
Host System 111
Host Processor(s) 200
220
Network Interface Controller 204
HT MAC 222
DMA Engine 224
DMA Engine 226
Ring to Qos Mapper 228
Mailbox Register 230
Engress DMA Routing 232
Ethernet MACs 234
236
Ethernet Port

FIG. 2B

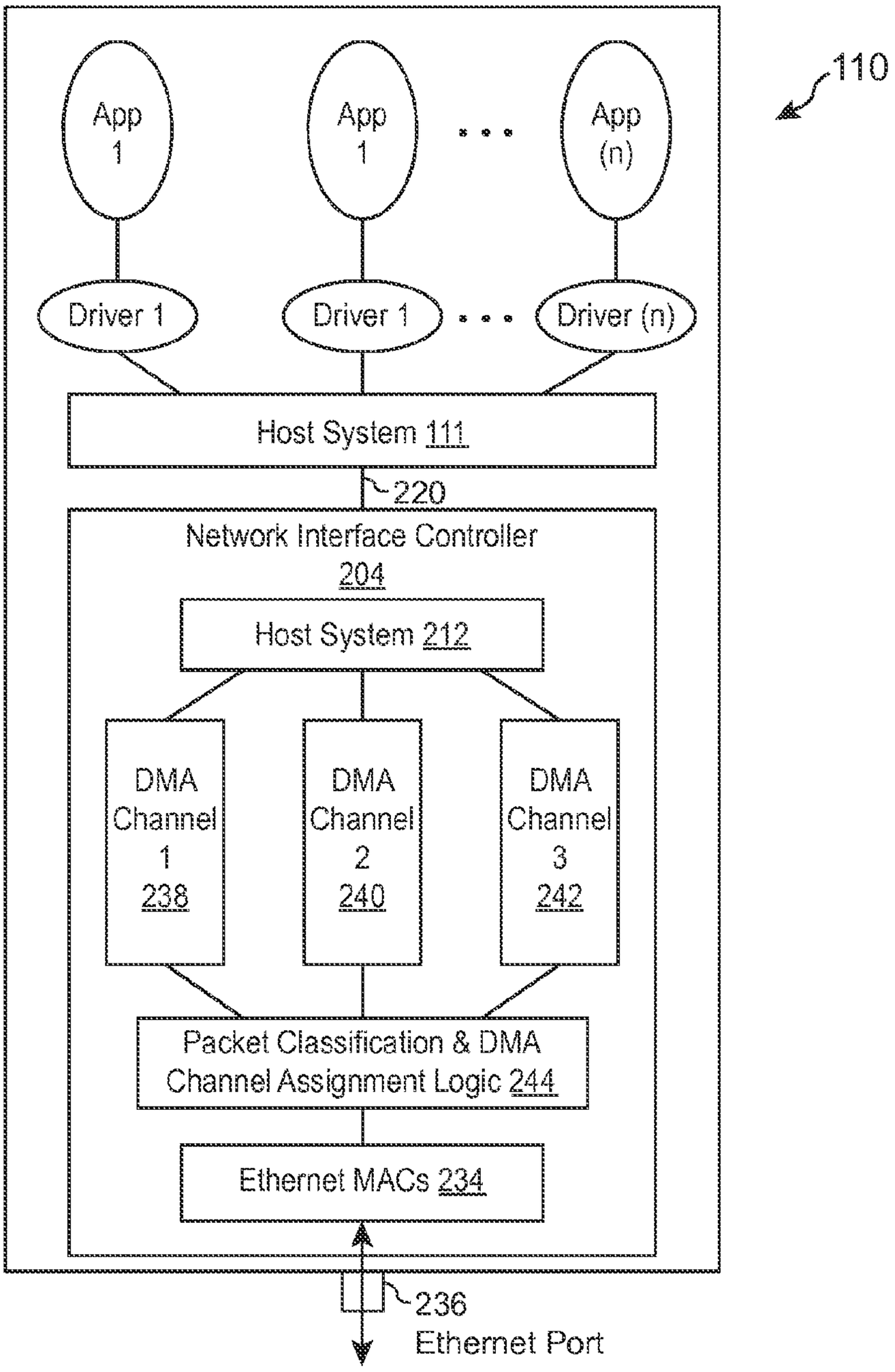

110
App 1
App 1
App (n)
Driver 1
Driver 1
Driver (n)
Host System 111
220
Network Interface Controller 204
Host System 212
DMA Channel 1 238
DMA Channel 2 240
DMA Channel 3 242
Packet Classification & DMA Channel Assignment Logic 244
Ethernet MACs 234
236
Ethernet Port

FIG. 2C

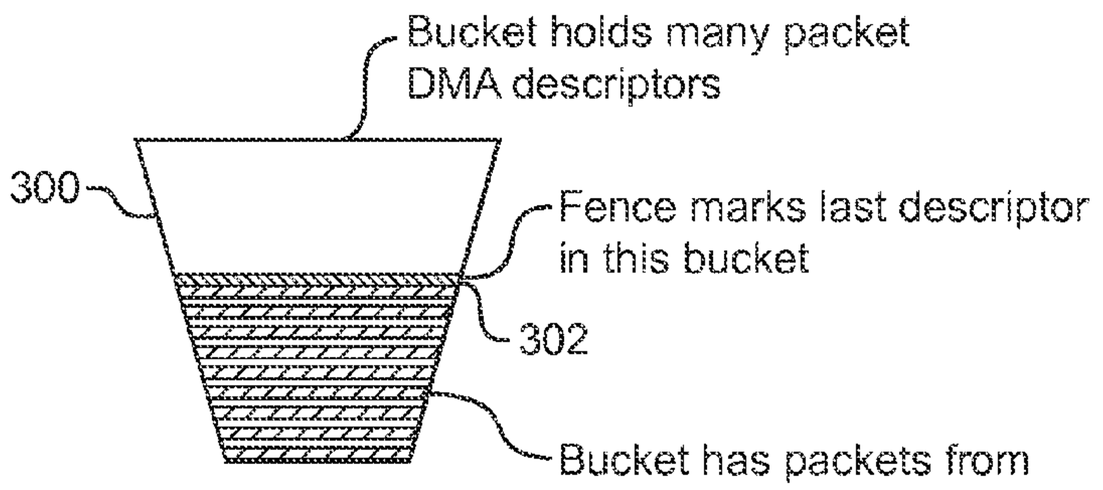

Bucket holds many packet DMA descriptors
300
Fence marks last descriptor in this bucket
302
Bucket has packets from many sessions

FIG. 3

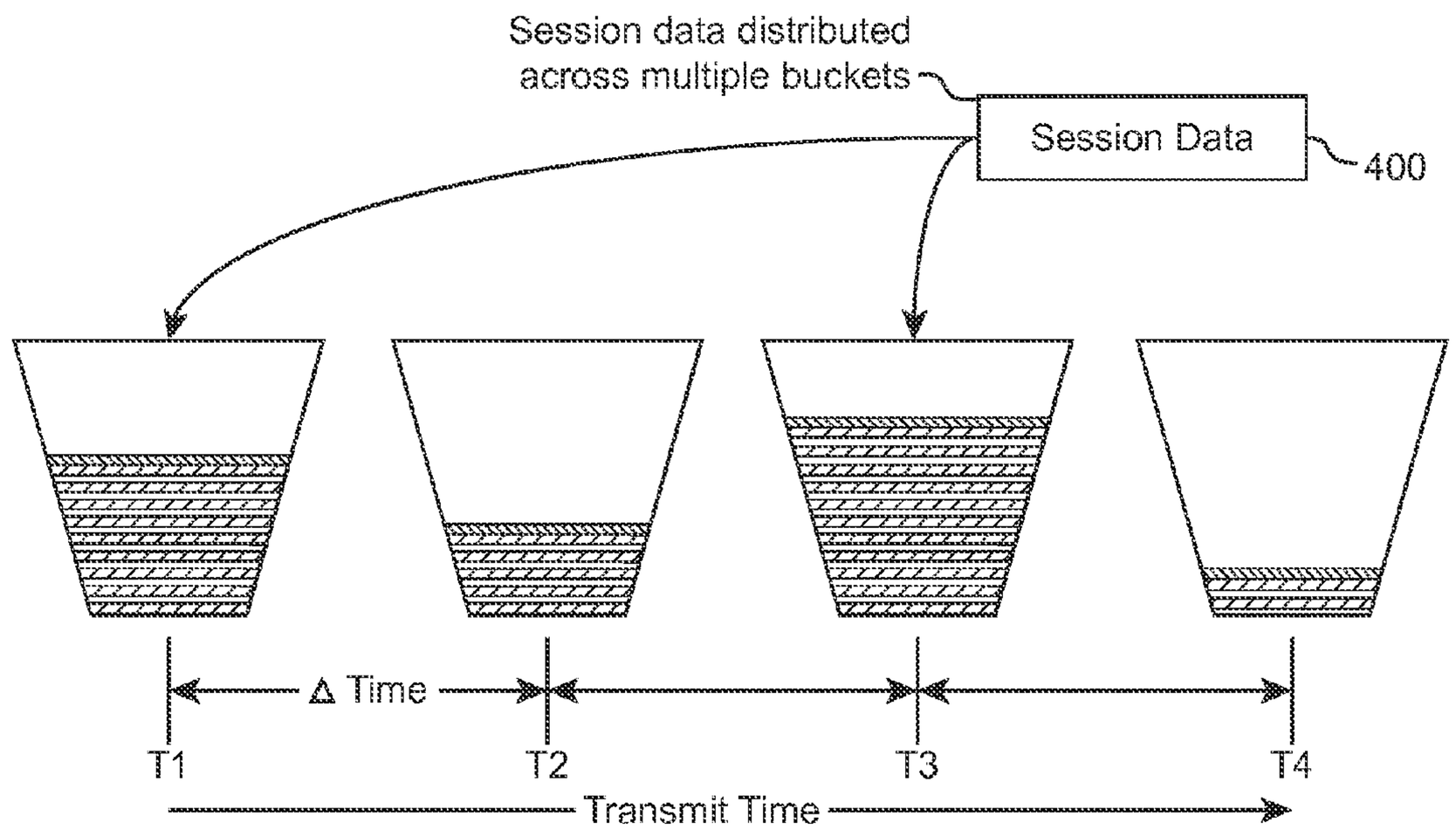

Session data distributed across multiple buckets
Session Data
400
Δ Time
T1
T2
T3
T4
Transmit Time

FIG. 4

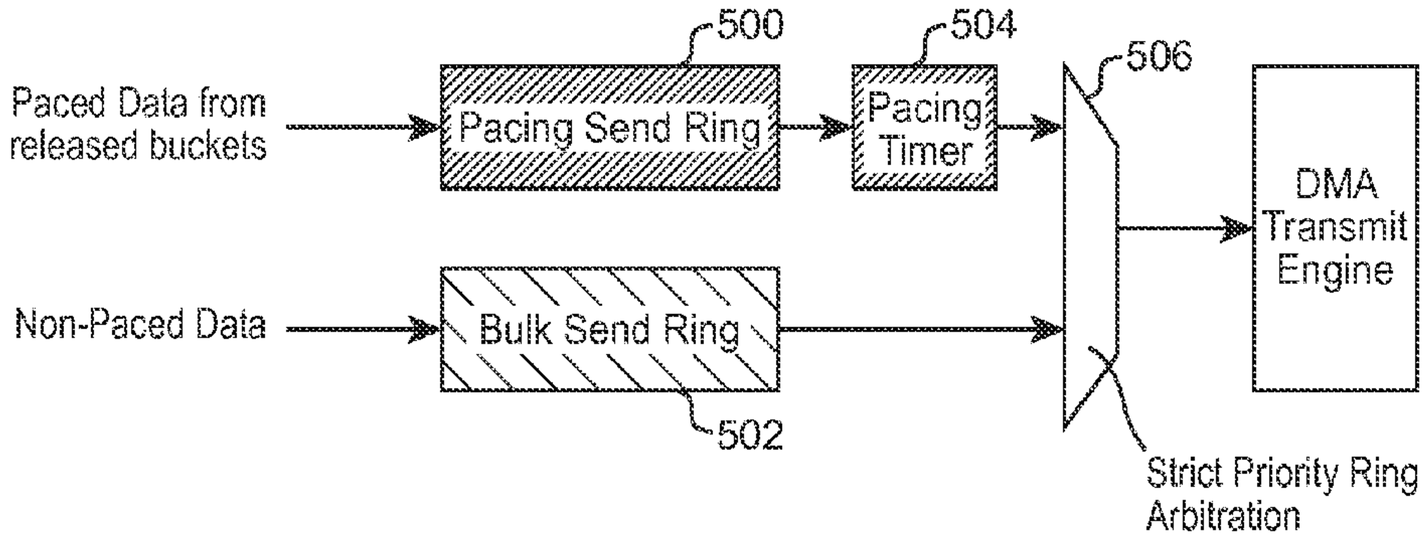

Paced Data from released buckets
Pacing Send Ring
500
Pacing Timer
504
506
DMA Transmit Engine
Non-Paced Data
Bulk Send Ring
502
Strict Priority Ring Arbitration

FIG. 5

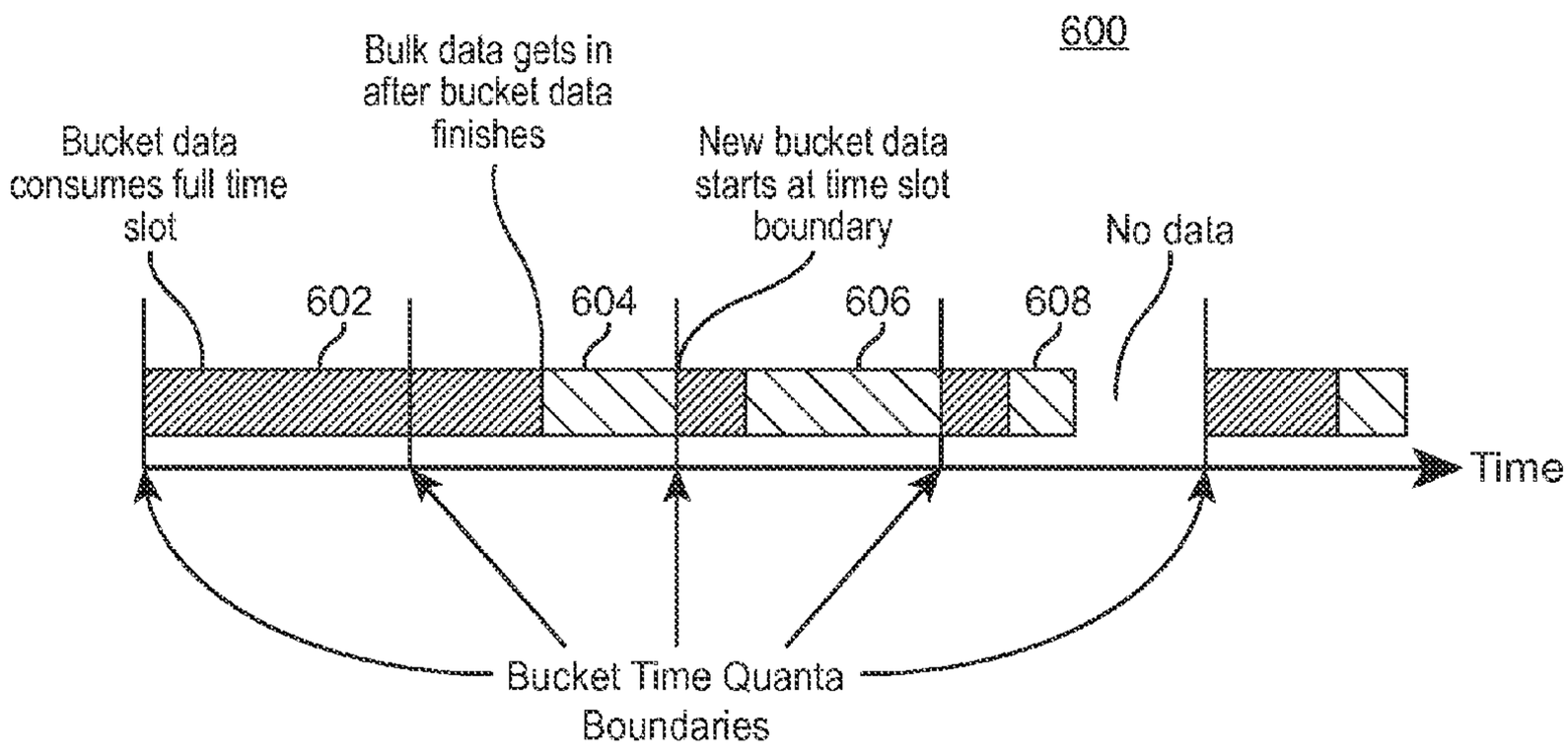

600
Bulk data gets in after bucket data finishes
Bucket data consumes full time slot
New bucket data starts at time slot boundary
No data
602
604
606
608
Time
Bucket Time Quanta Boundaries

FIG. 6

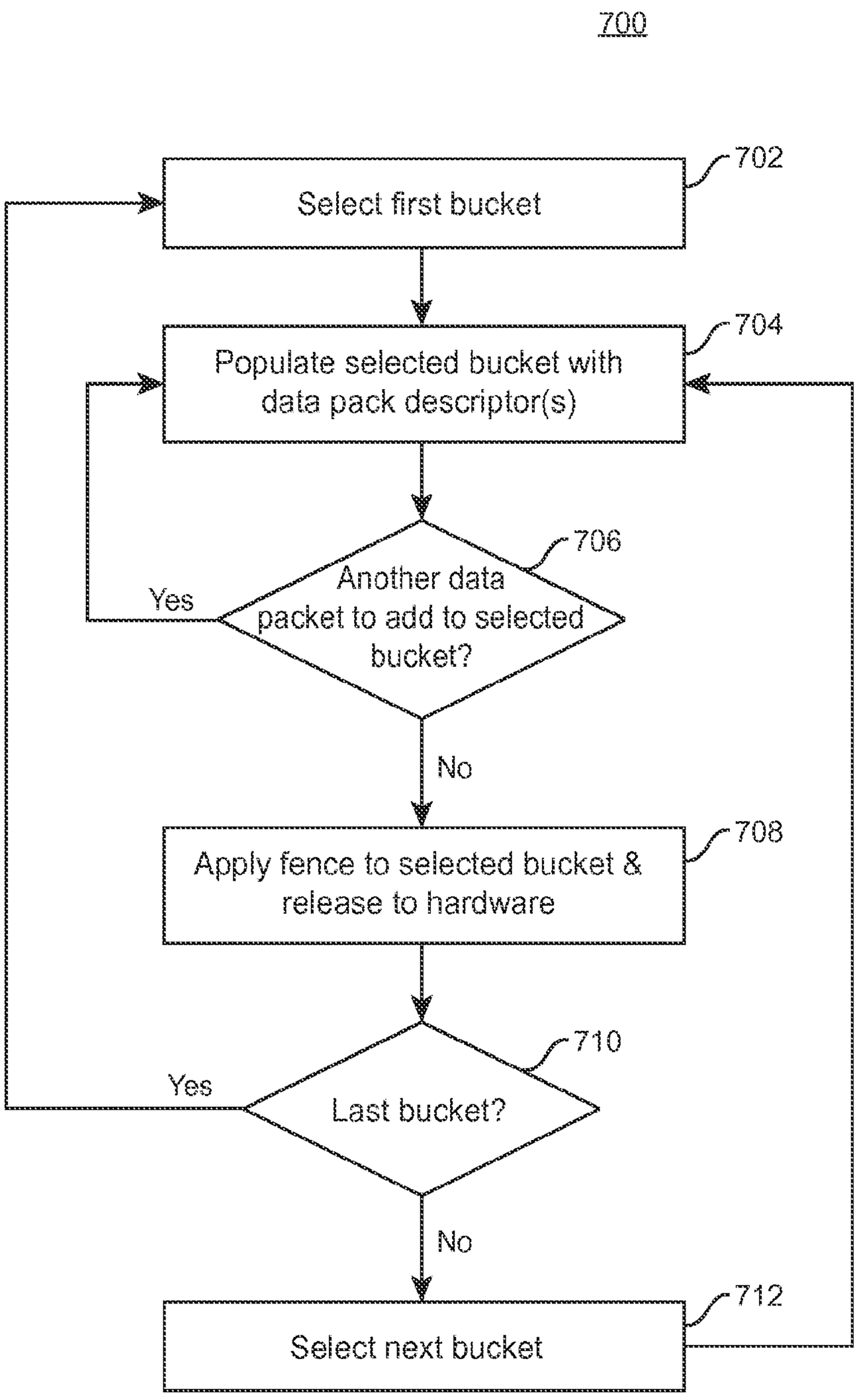

700
Select first bucket
702
Populate selected bucket with data pack descriptor(s)
704
Another data packet to add to selected bucket?
706
Yes
No
Apply fence to selected bucket & release to hardware
708
Last bucket?
710
Yes
No
Select next bucket
712

FIG. 7

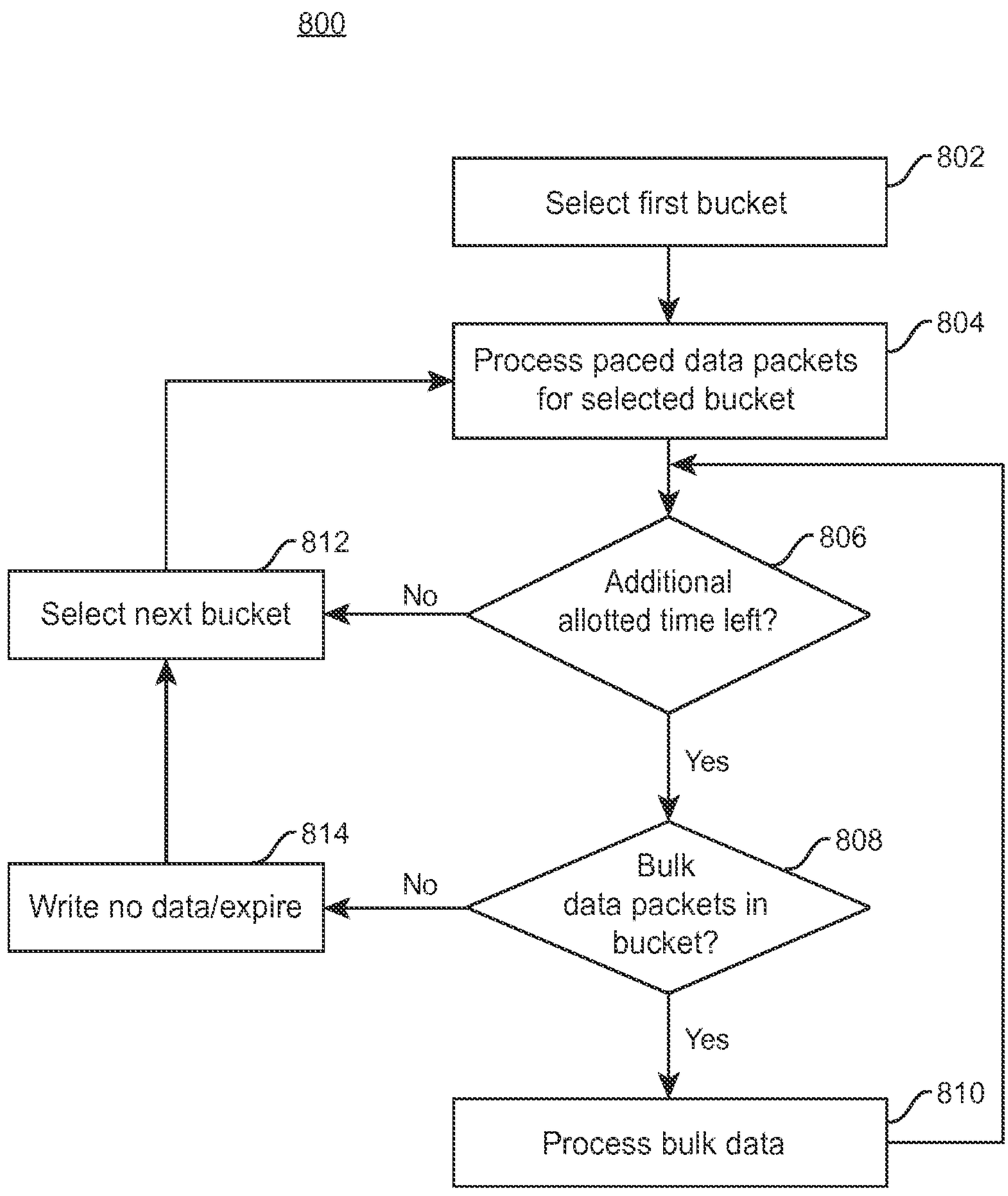

800
Select first bucket
802
Process paced data packets for selected bucket
804
Additional allotted time left?
806
No
Select next bucket
812
Yes
Bulk data packets in bucket?
808
No
Write no data/expire
814
Yes
Process bulk data
810

FIG. 8

TRANSMIT RATE PACING OF LARGE NETWORK TRAFFIC BURSTS TO REDUCE JITTER, BUFFER OVERRUN, WASTED BANDWIDTH, AND RETRANSMISSIONS

FIELD

This technology relates to a system and method for pacing network traffic between network devices.

BACKGROUND

It is very common for a modern server to transmit large blocks of data in one burst to a single destination where the network path to the destination has much lower bandwidth than the really large bandwidth of the server's path within the data center and the first few hops along that path. Often these bursts are in response to a request for data, and the data is read from a storage medium (e.g., a disk) in a large block to help amortize the cost of the read operation. The large chunk of data is then dumped by the server's OS into the network at full speed, adding latency for other traffic following some portion of the same path through the network. This latency is unnecessary since other traffic could easily have been interleaved with the packets of the burst if the packets of the burst were spaced in time to match the true bandwidth of the full path to the destination. Further, such bursts can result in loss of some of the burst data due to overruns in of the buffering in the network path to the destination. Such losses necessitate retransmission of some of the large chunk of data—effectively reducing the gains that were hoped to be achieved by the batching of the read operation into a large chunk and reducing the overall throughput of the server. If the packets in these bursts were, instead, spread out in time at a pace matching the full network path data rate, both of these problems are easily solved

What is needed is a system and method which overcomes these disadvantages.

SUMMARY

In an aspect, a method for pacing data packets from one or more sessions comprises selecting, at a software component of a network traffic management device, a first bucket having a first predetermined transmit time. The method comprises populating one or more selected data packet descriptors associated with one or more corresponding data packets in the first bucket. The method comprises releasing the first bucket to a hardware component of the network traffic management device, wherein the hardware component processes the one or more data packet descriptors of the first bucket for the first predetermined transmit time.

In an aspect, a processor readable medium having stored thereon instructions for pacing data packets from one or more sessions, comprising machine executable code which when executed by at least one processor and/or network interface a network traffic management device to perform a method comprising selecting a first bucket having a first predetermined transmit time; populating one or more selected data packet descriptors associated with one or more corresponding data packets in the first bucket; releasing the first bucket to a hardware component of the network traffic management device, wherein the hardware component processes the one or more data packet descriptors of the first bucket for the first predetermined transmit time.

In an aspect, a network traffic management device comprises a memory containing non-transitory machine readable medium comprising machine executable code having stored thereon instructions for pacing data packets from one or more sessions. A network interface configured to communicate with one or more servers over a network. A processor coupled to the network interface and the memory, the processor configured to execute the code which causes the processor to perform, with the network interface, a method comprising: selecting a first bucket having a first predetermined transmit time; populating one or more selected data packet descriptors associated with one or more corresponding data packets in the first bucket; releasing the first bucket to a hardware component of the network traffic management device, wherein the hardware component processes the one or more data packet descriptors of the first bucket for the first predetermined transmit time.

In one or more of the above aspects, the method performs comprises selecting, at the software component, a second bucket having a second predetermined transmit time that is the same as the first transmit time of the first bucket; populating one or more selected data packet descriptors associated with one or more corresponding data packets in the second bucket; releasing the second bucket to the hardware component of the network traffic management device, wherein the hardware component processes the one or more data packet descriptors of the second bucket for the second predetermined transmit time quanta.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a diagram of an example system environment that includes a network traffic management device in accordance with an aspect of the present disclosure.

FIG. **2**A is a block diagram of the network traffic management device in accordance with an aspect of the present disclosure.

FIG. **2**B illustrates a block diagram of the network interface in accordance with an aspect of the present disclosure.

FIG. **2**C illustrates further details of the network traffic management device in accordance with an aspect of the present disclosure.

FIG. **3** illustrates an example transmit ring or bucket in accordance with an aspect of the present disclosure.

FIG. **4** illustrates a time line of an example transmit ring or bucket in accordance with an aspect of the present disclosure.

FIG. **5** illustrates a block diagram of hardware based time enforcement in performed by a high speed bridge (HSB) priority mechanism in accordance with an aspect of the present disclosure.

FIG. **6** illustrates an example transmit time line in accordance with an aspect of the present disclosure.

FIG. **7** illustrates a flow chart of the software implementation of populating buckets with data packet descriptors in accordance with an aspect of the present disclosure.

FIG. **8** illustrates a flow chart of the software implementation of populating buckets with data packet descriptors in accordance with an aspect of the present disclosure.

While these examples are susceptible of embodiment in many different forms, there is shown in the drawings and will herein be described in detail preferred examples with the understanding that the present disclosure is to be considered as an exemplification and is not intended to limit the broad aspect to the embodiments illustrated.

DETAILED DESCRIPTION

FIG. **1** is a diagram of an example system environment that includes a network traffic management device in accordance

with an aspect of the present disclosure. The example system environment **100** includes one or more Web and/or non Web application servers **102** (referred generally as "servers"), one or more client devices **106** and one or more network traffic management devices **110**, although the environment **100** can include other numbers and types of devices in other arrangements. The network traffic management device **110** is coupled to the servers **102** via local area network (LAN) **104** and client devices **106** via a wide area network **108**. Generally, client device requests are sent over the network **108** to the servers **102** which are received or intercepted by the network traffic management device **110**.

Client devices **106** comprise network computing devices capable of connecting to other network computing devices, such as network traffic management device **110** and/or servers **102**. Such connections are performed over wired and/or wireless networks, such as network **108**, to send and receive data, such as for Web-based requests, receiving server responses to requests and/or performing other tasks. Non-limiting and non-exhausting examples of such client devices **106** include personal computers (e.g., desktops, laptops), tablets, smart televisions, video game devices, mobile and/or smart phones and the like. In an example, client devices **106** can run one or more Web browsers that provide an interface for operators, such as human users, to interact with for making requests for resources to different web server-based applications and/or Web pages via the network **108**, although other server resources may be requested by client devices.

The servers **102** comprise one or more server network devices or machines capable of operating one or more Web-based and/or non Web-based applications that may be accessed by other network devices (e.g. client devices, network traffic management devices) in the environment **100**. The servers **102** can provide web objects and other data representing requested resources, such as particular Web page(s), image(s) of physical objects, JavaScript and any other objects, that are responsive to the client devices' requests. It should be noted that the servers **102** may perform other tasks and provide other types of resources. It should be noted that while only two servers **102** are shown in the environment **100** depicted in FIG. **1**B, other numbers and types of servers may be utilized in the environment **100**. It is contemplated that one or more of the servers **102** may comprise a cluster of servers managed by one or more network traffic management devices **110**. In one or more aspects, the servers **102** may be configured implement to execute any version of Microsoft® IIS server, RADIUS server, DIAMETER server and/or Apache® server, although other types of servers may be used.

Network **108** comprises a publicly accessible network, such as the Internet, which is connected to the servers **102**, client devices **106**, and network traffic management devices **110**. However, it is contemplated that the network **108** may comprise other types of private and public networks that include other devices. Communications, such as requests from clients **106** and responses from servers **102**, take place over the network **108** according to standard network protocols, such as the HTTP, UDP and/or TCP/IP protocols, as well as other protocols. As per TCP/IP protocols, requests from the requesting client devices **106** may be sent as one or more streams of data packets over network **108** to the network traffic management device **110** and/or the servers **102**. Such protocols can be utilized by the client devices **106**, network traffic management device **110** and the servers **102** to establish connections, send and receive data for existing connections, and the like.

Further, it should be appreciated that network **108** may include local area networks (LANs), wide area networks (WANs), direct connections and any combination thereof, as well as other types and numbers of network types. On an interconnected set of LANs or other networks, including those based on differing architectures and protocols. Network devices such as client devices, **106**, servers **102**, network traffic management devices **110**, routers, switches, hubs, gateways, bridges, cell towers and other intermediate network devices may act within and between LANs and other networks to enable messages and other data to be sent between network devices. Also, communication links within and between LANs and other networks typically include twisted wire pair (e.g., Ethernet), coaxial cable, analog telephone lines, full or fractional dedicated digital lines including T1, T2, T3, and T4, Integrated Services Digital Networks (ISDNs), Digital Subscriber Lines (DSLs), wireless links including satellite links and other communications links known to those skilled in the relevant arts. Thus, the network **108** is configured to handle any communication method by which data may travel between network devices.

LAN **104** comprises a private local area network that allows communications between the one or more network traffic management devices **110** and one or more servers **102** in the secured network. It is contemplated, however, that the LAN **104** may comprise other types of private and public networks with other devices. Networks, including local area networks, besides being understood by those skilled in the relevant arts, have already been generally described above in connection with network **108** and thus will not be described further.

As shown in the example environment **100** depicted in FIG. **1**B, the one or more network traffic management devices **110** is interposed between client devices **106** with which it communicates with via network **108** and servers **102** with which it communicates with via LAN **104**. In particular to the present disclosure, the network traffic management device **110** operates in conjunction with a clustered multi-processing (CMP) system which includes one or more network traffic management devices **110**, each of which having one or more cores or processors **200** (FIG. **2**A). Generally, the network traffic management device **110** manages network communications, which may include one or more client requests and server responses, via the network **108** between the client devices **106** and one or more of the servers **102**. In any case, the network traffic management device **110** may manage the network communications by performing several network traffic related functions involving the communications. Some functions include, but are not limited to, load balancing, access control, and validating HTTP requests using JavaScript code that are sent back to requesting client devices **106**.

FIG. **2**A is a block diagram of the network traffic management device in accordance with an aspect of the present disclosure. As shown in FIG. **2**A, the example network traffic management device **110** includes one or more device processors or cores **200**, one or more device I/O interfaces **202**, one or more network interfaces **204**, and one or more device memories **206**, which are coupled together by one or more bus **208**. It should be noted that the network traffic management device **110** can be configured to include other types and/or numbers of components and is thus not limited to the configuration shown in FIG. **2**A.

Device processor **200** of the network traffic management device **110** comprises one or more microprocessors configured to execute computer/machine readable and executable instructions stored in the device memory **206**. Such instructions, when executed by one or more processors **200**, imple-

ment general and specific functions of the network traffic management device **110**, including the inventive process described in more detail below. It is understood that the processor **200** may comprise other types and/or combinations of processors, such as digital signal processors, micro-controllers, application specific integrated circuits ("ASICs"), programmable logic devices ("PLDs"), field programmable logic devices ("FPLDs"), field programmable gate arrays ("FPGAs"), and the like. The processor **200** is programmed or configured according to the teachings as described and illustrated herein.

Device I/O interfaces **202** comprise one or more user input and output device interface mechanisms. The interface may include a computer keyboard, mouse, display device, and the corresponding physical ports and underlying supporting hardware and software to enable the network traffic management device **110** to communicate with other network devices in the environment **100**. Such communications may include accepting user data input and providing user output, although other types and numbers of user input and output devices may be used. Additionally or alternatively, as will be described in connection with network interface **204** below, the network traffic management device **110** may communicate with the outside environment for certain types of operations (e.g. smart load balancing) via one or more network management ports.

Network interface **204** comprises one or more mechanisms that enable the network traffic management device **110** to engage in network communications over the LAN **104** and the network **108** using one or more of a number of protocols, such as TCP/IP, HTTP, UDP, RADIUS and DNS. However, it is contemplated that the network interface **204** may be constructed for use with other communication protocols and types of networks. Network interface **204** is sometimes referred to as a transceiver, transceiving device, or network interface card (NIC), which transmits and receives network data packets over one or more networks, such as the LAN **104** and the network **108**. In an example, where the network traffic management device **110** includes more than one device processor **200** (or a processor **200** has more than one core), each processor **200** (and/or core) may use the same single network interface **204** or a plurality of network interfaces **204**. Further, the network interface **204** may include one or more physical ports, such as Ethernet ports, to couple the network traffic management device **110** with other network devices, such as servers **102**. Moreover, the interface **204** may include certain physical ports dedicated to receiving and/or transmitting certain types of network data, such as device management related data for configuring the network traffic management device **110** or client request/server response related data.

Bus **208** may comprise one or more internal device component communication buses, links, bridges and supporting components, such as bus controllers and/or arbiters. The bus **208** enables the various components of the network traffic management device **110**, such as the processor **200**, device I/O interfaces **202**, network interface **204**, and device memory **206**, to communicate with one another. However, it is contemplated that the bus **208** may enable one or more components of the network traffic management device **110** to communicate with one or more components in other network devices as well. Example buses include HyperTransport, PCI, PCI Express, InfiniBand, USB, Firewire, Serial ATA (SATA), SCSI, IDE and AGP buses. However, it is contemplated that other types and numbers of buses may be used, whereby the particular types and arrangement of buses will depend on the particular configuration of the network traffic management device **110**.

Device memory **206** comprises computer readable media, namely computer readable or processor readable storage media, which are examples of machine-readable storage media. Computer readable storage/machine-readable storage media may include volatile, nonvolatile, removable, and non-removable media implemented in any method or technology for storage of information. Examples of computer readable storage media include RAM, BIOS, ROM, EEPROM, flash/firmware memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the information, which can be accessed by a computing or specially programmed network device, such as the network traffic management device **110**.

Such storage media includes computer readable/processor-executable instructions, data structures, program modules, or other data, which may be obtained and/or executed by one or more processors, such as device processor **200**. Such instructions, when executed, allow or cause the processor **200** to perform actions, including performing the inventive processes described below. The memory **206** may contain other instructions relating to the implementation and operation of an operating system for controlling the general operation and other tasks performed by the network traffic management device **110**.

FIG. **2**B illustrates a block diagram of the network interface in accordance with an aspect of the present disclosure. In particular, FIG. **2**B shows the DMA processes used by network interface **204** for using multiple independent DMA channels with corresponding multiple applications, where each application has its own driver, and for sending packets.

As illustrated in FIG. **2**B, the host system **111** can send a network data packet stored in host memory **212** to the network **108** via network interface controller **204** and Ethernet port **236**. A send DMA operation is performed when the host system **111** uses a DMA channel to move a block of data from host memory **22** to a network interface controller peripheral (not shown) via network **108**. To perform a send DMA operation, the host processor **200** places the target network data packet into DMA packet buffer **216** and creates a DMA send descriptor (not shown separately) in send DMA descriptor rings **218**. The DMA send descriptor is jointly managed by the host system **111** and the network interface controller **204**. The DMA send descriptor includes an address field and length field. The address field points to the start of the target network data packet in DMA packet buffer **216**. The length field declares how many bytes of target data are present in the DMA packet buffer **216**. The DMA send descriptor also has a set of bit flags (not shown) used to signal additional target data control and status information.

By way of example only, return DMA descriptor rings and send DMA descriptor rings **218** can be physically in the same hardware memory blocks functioning as return and send DMA rings, respectively, at different times. Alternatively, separate and distinct memory blocks within host memory **212** DMA memory resources **214** may be reserved for each return DMA descriptor rings and send DMA descriptor rings **218**.

Host system **111** places the send descriptor on the send DMA descriptor rings **218** in host system memory **212**. The host processor **200** determines the QoS of the network packet to be transferred to the network **108** and moves the network packet to the appropriate DMA packet buffer **216** and places the descriptor on the appropriate descriptor rings 1-4 in send DMA descriptor rings **218**. The descriptor ring in send DMA descriptor rings **218** is chosen by the host system **111** which selects the DMA channel, its associated peripheral, and the

QoS level within the DMA channel. Send descriptors created by host system **111** in send DMA descriptor rings **218** can be of variable types, where each descriptor type can have a different format and size. The send DMA descriptor rings **218** are capable of holding descriptors of variable type.

The host processor **200** writes one or more mailbox registers **230** of the network interface controller **204** to notify the network interface controller **204** that the packet is ready. In performing this notification, the host processor **200** performs a write operation to a memory mapped network interface controller register (mailbox register **230**). The host processor **200** can report the addition of multiple descriptors onto the send DMA ring in a single update, or alternatively, in multiple updates.

The appropriate packet DMA engine within DMA engine **224** is notified that the packet is ready. The packet DMA engine **224** can be selected from available DMA channels, or if a specific application has a dedicated DMA channel, the associated packet DMA engine **224** for that channel is used. The DMA engine **224** retrieves the DMA descriptor from the send DMA descriptor rings **218**. When multiple descriptors are outstanding in the send DMA descriptor rings **218**, the DMA Engine **224** may retrieve more than one descriptor. Retrieving multiple descriptors at a time maximizes bus bandwidth and hardware efficiency. The DMA engine **224** is capable of receiving and processing send descriptors of variable type, format, and size.

As outlined above, the packet DMA engine **224** monitors the progress of the host DMA operations via a set of mailbox registers **230**. Each packet DMA engine **224** supports its own set of mailbox registers **230**. The mailbox registers **230** reside in a mapped address space of the network interface controller **204**. When appropriate, the host processor **200** accesses the mailbox registers **230** by performing memory mapped read and write transactions to the appropriate target address. The mailbox registers **230** also contain ring status information for the Ring to QoS Mapper **228**. In this send DMA example, the packet DMA engine **224** reads the send descriptor, performs the DMA operation defined by it, and reports to the host system **111** that the DMA operation is complete. During the DMA operation, data is received from one or more CPU Bus read transactions (e.g., HyperTransport or PCI Express read transactions).

DMA scheduler **226** chooses packets out of packet buffers **216** based upon the priority of the queued network data packets and schedules the transfer to the appropriate packet DMA engine **224**. For clarity and brevity, only a single packet buffer, a single DMA scheduler, and DMA engine are shown in FIG. **2**B, but it should be understood that additional packet buffers, DMA schedulers, and DMA engines supporting the independent DMA channels 1-n and associated applications App(1)-App(n) can be included in network interface controller **204**.

The packet buffers **216** are selected based on the novel scheme (discussed below) using DMA scheduler **226**. The DMA scheduler **226** selects which descriptor ring 1-4 out of return DMA descriptor rings (also referred to as return DMA rings, or send rings) within DMA memory resources **212** to service and the matching packet buffer **216** is accessed for a single packet. The scheduling process is then repeated for the next packet.

Each network packet retrieved from a packet buffer **216** is routed to the appropriate DMA channel controlled by the respective packet DMA engine such as the packet DMA engine **224**. The DMA channel segments the network packet for delivery to host memory **212** via several, smaller, HyperTransport packets. These HyperTransport packets are interleaved with HyperTransport packets from the other DMA channels in the network interface controller **204**.

Ring to QoS Mapper **228** examines the assigned send DMA ring in send DMA descriptor rings **218** and receives packet data and packet control information from the packet DMA engine **224**. Using the control information, the Ring to QoS Mapper **228** stamps the appropriate QoS onto the network data packet, thereby allowing host system **111** to send the network data packet back to the network **108**. For example, using the control information, the Ring to QoS Mapper **228** can create and prepend a HiGig header to the packet data.

An egress DMA routing interface **232** arbitrates access to the network for DMA send packets. When a Ring to QoS Mapper **228** has a network packet ready to send, the egress DMA routing interface **232** arbitrates its access to the Ethernet port **236** and routes the packet to the correct interface if there is more than one present in the network interface controller **204**. The egress DMA routing interface **232** behaves like a crossbar switch and monitors its attached interfaces for available packets. When a packet becomes available, the egress DMA routing interface **232** reads the packet from the selected ring to QoS mapper **228** and writes it to the destination interface. The egress DMA routing interface **232** moves complete packets to Ethernet MACs **234**. When multiple sources are contending for egress DMA routing interface **232**, the egress DMA routing interface **232** uses a weighted arbitration scheme as discussed in more detail below.

The network interface controller **204** provides DMA services to a host complex such as the host system **111** on behalf of its attached I/O devices such as the Ethernet port **236**. DMA operations involve the movement of data between the host memory **212** and the network interface controller **204**. The network interface controller **204** creates and manages HyperTransport or other types of CPU Bus read/write transactions targeting host memory **22**. Data transfer sizes supported by DMA channels maintained by various components of application delivery controller **110** are much larger than the maximum HyperTransport or CPU bus transaction size. The network interface controller **204** segments single DMA operations into multiple smaller CPU Bus or HyperTransport transactions. Additionally, the network interface controller **204** creates additional CPU bus or HyperTransport transactions to support the transfer of data structures between the network interface controller **204** and host memory **212**.

FIG. **2**C illustrates further details of the network traffic management device in accordance with an aspect of the present disclosure. In particular, the network traffic management device **110** is shown handling a plurality of independent applications App(**1**)-App(n) being executed by one or more processors (e.g., host processor **200**) in host system **111**. Each application in the plurality of applications App(1)-App(n) has its own respective application driver shown as Driver 1, Driver 2, . . . , Driver 'n' associated with the respective application, where the index n denotes an unlimited number of executing applications and drivers. Applications App(1)-App(n) send and receive data packets from and to the network **108** (and/or LAN **104**), respectively, using respective DMA channels (e.g., DMA channels 1-n). DMA channels 1-n are uniquely assigned to individual applications out of App(1)-App(n). In this example, drivers 1-n manage access to respective DMA channels 1-n and do not require knowledge of each other or a common management database or entity (e.g., a hypervisor). By way of example only, each of applications App(1)-App(n) can be independent instances of different applications, or alternatively, may be independent instances

of the same application, or further, may be different operating systems supported by different processors in host system **111** (e.g., host processor **200**).

DMA channels 1-n each have unique independent resources allotted to them, for example, a unique PCI bus identity including a configuration space and base address registers, an independent view of host system memory **212**, a unique set of DMA descriptor ring buffers, a unique set of packet buffers **216**, unique DMA request/completion signaling (through interrupts or polled memory structures), and other resources. Each of DMA channels 1-n is unique and independent thereby permitting management by separate unique drivers 1-n.

The network interface controller **204** classifies received packets to determine destination application selected from applications App(1)-App(n) and thereby selects the matching DMA channel to deliver the packet to the corresponding application. By way of example only, packet classification includes reading packet header fields thereby permitting application identification. Further by way of example only, packet classification includes hash calculation for distribution of packets across multiple instances of the same application, and/or reading a cookie stored, for example, in the network interface controller **204** associated with the application and the received network packet.

In general, the present disclosure utilizes hardware and software in the network traffic management device when pacing network traffic being sent to another network device (e.g. client, server). The purpose is for the network traffic management device to pace delivery of session data to the client to match the rate at which the client consumes the data, which adds value for mobile clients and networks. The network traffic management device utilizes a transmit time calendar having a plurality of quanta of transmit times. For example, one quanta may be 1 microsecond, although other time durations are contemplated.

FIG. **3** illustrates an example transmit ring or bucket in accordance with an aspect of the present disclosure. As shown in FIG. **3**, the bucket **300** is a software based data construct which holds a plurality of data packet DMA descriptors which point to the pending transmit packets and has packets from a plurality of various, different sessions. The bucket **300** has a fence **302** which marks the last descriptor in the bucket. Software determines what data packets to send and when to send those data packets. The software communicates with the hardware component wherein the software enhances the data structures to communicate a time component with regard to when the selected data packets are to be sent. In particular, the software component divides session data into multiple maximum segment sized (MSS) or smaller sized packets. The software component distributes the packets in buckets separated by fixed transmit times, wherein the buckets are then handed off to the hardware component, such as a DMA ring, to handle the delivery of the data packets populated in the buckets, by processing the buckets themselves.

FIG. **4** illustrates a time line of an example transmit ring or bucket in accordance with an aspect of the present disclosure. As can be seen in FIG. **4**, the transmit time calendar utilizes a plurality of buckets **300** where session data **400**, such as data packet DMA descriptors is populated into the buckets **300**. Each bucket has a predetermined size and adjacent buckets are separate by a transmit time or quanta that is fixed (shown as T1-T4). It should be noted the time calendar is configured to have a finite number of buckets, wherein the calendar effectively wraps around to the first bucket after the last bucket has been handled. In addition, multiple packet descriptors can be placed in the same bucket to increase bandwidth. The software component may decide whether to pace a data packet, as opposed to being bulk data (FIG. **5**) based on size, type of traffic the data packet is associated with, QoS parameters (based on flow) and the like.

In an aspect, one or more buckets **300** can be skipped for distribution, by the software component, to increase intra-packet spacing. One way for determining how many buckets to skip before placing a data packet descriptor in a bucket **300** can be based on the type or class of application in which the data is delivered (e.g. video streaming). For example, video streaming applications would benefit from a constant rate of data delivery. Another way of determining (based on speed) would be using TCP congestion control algorithms which calculate how many packets are to be sent over time. The software does not have to hunt for holes where packet descriptors can be inserted. Instead, the software component need only to drop packet descriptors into the one or more buckets **300**.

Once a bucket **300** is populated by the software component, the software component releases the bucket to the hardware component, such as the network interface **204**. In particular, the bucket contents are written into the network interface's **204** DMA ring **218** en-mass. In an aspect, the hardware component can determine whether the size of a particular data packet in a bucket is of a threshold size such that the data packet can be processed within the allotted time quanta.

A gross timer is applied by the hardware component in writing the packets into the DMA ring **218**, wherein the poll loop time of the processor **200** of the network traffic management device **110** is used as the gross timer. The sum of released bucket time quanta's must excel poll loop time.

The fence **304** marking at the end of a particular bucket **300** serves as a timing boundary at the end of the bucket to allow the hardware component to do precise bucket to bucket timing.

FIG. **5** illustrates a block diagram of hardware based time enforcement in performed by a high speed bridge (HSB) priority mechanism in accordance with an aspect of the present disclosure. As shown in FIG. **5**, data packets that are written to buckets that are released to the hardware are received in a pacing send ring **500**. In contrast, non-paced data (data not written to buckets) are received in a bulk send ring **502**.

In an aspect, the pacing send ring **500** is given higher priority in terms of being sent to the ring arbitrator **506**. Thus, the bulk traffic in the bulk send ringer **502** advances to the arbitrator **506** when the paced traffic (from the pacing send ring **500**) is blocked or is absent. A pacing timer **504** in coupled to the pacing send ring **500** and the arbitrator **506**, wherein the pacing timer is reset at the start of a new bucket **300**. The pacing timer **504** also blocks traffic at bucket boundaries until the bucket quanta time expires.

FIG. **6** illustrates an example transmit time line in accordance with an aspect of the present disclosure. As shown in the time line **600**, the hardware component consumes the data packets in the first bucket **602** for the entire time quanta which ends at boundary A. In the second bucket **604**, it is shown that the hardware component finishes writing the paced data packets such that additional time remains for the allotted time. The hardware component thereafter processes bulk data for the remaining amount of time in the time quanta until reaching boundary B. The same occurs in bucket **606**. In bucket **608**, there is a period of time where no data is written, which can result in no bulk data or paced data being available for writing to the DMA engine.

FIG. **7** illustrates a flow chart of the software implementation of populating buckets with data packet descriptors in

accordance with an aspect of the present disclosure. As shown in FIG. **7**, the process **700** occurs when a first bucket of a plurality of buckets is selected by the application module **210** of the network traffic management device (Block **702**). The application module **210** thereafter populates the first bucket with one or more data packet descriptors associated with data packets to be released to the network interface **204** (Block **704**). The application module **210** thereafter determines whether there are additional paced or bulk data packets that can be added to the current selected bucket (Block **706**). If so, the process repeats back to Block **704**. If not, the process proceeds to Block **708**.

Once the application module **210** has populated the bucket with the last data packet, the application module **210** applies a fence which represents the last data packet for that bucket (Block **708**). The application module **210** thereafter determines if the selected bucket is the last bucket in the time calendar (Block **710**). If so, the application module **210** effectively wraps around the time calendar and selects the first bucket (Block **702**). If not, the application module **210** selects the next bucket in the time calendar (Block **712**), wherein the process proceeds back to Block **704**.

FIG. **8** illustrates a flow chart of the software implementation of populating buckets with data packet descriptors in accordance with an aspect of the present disclosure. As shown in FIG. **8**, the process **800** occurs when a first bucket of a plurality of buckets is selected by the network interface **204** of the network traffic management device (Block **802**). The network interface **204** thereafter processes paced data packet (s) for the selected bucket (Block **804**). The network interface **204** thereafter determines whether additional time is available for the selected bucket (Block **806**). If so, the network interface **204** determines if bulk data packets are present in the bucket (Block **808**). If so, the network interface **204** processes the bulk data for the bucket (Block **810**). The process then repeats back to Block **806**.

If no additional allotted time is left, the network interface **204** selects the next bucket **812** and the process proceeds back to Block **804**. Referring back to Block **808**, if no bulk data is available for processing for a particular bucket, the network interface **204** does not write any data packets and allows the remaining time for the bucket quanta to expire (Block **814**). The process then proceeds to Block **812**.

Having thus described the basic concepts, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the examples. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed system and/or processes to any order except as may be specified in the claims. Accordingly, the system and method is limited only by the following claims and equivalents thereto.

What is claimed is:

1. A method for transmitting data packets at an optimized rate, the method comprising:

populating, by the network traffic management computing device, a plurality of buckets with one or more selected data packet descriptors associated with one or more corresponding ones of a subset of a plurality of data packets to be transmitted as paced and another subset of the data packets to be transmitted as bulk;

releasing, by the network traffic management computing device, one of the buckets to a hardware component comprising a pacing send ring and a bulk send ring, the releasing comprising writing one or more of the subset of the data packets to the pacing send ring and one or more of the another subset of the data packets to the bulk send ring; and

transmitting, by the network traffic management computing device, the one or more of the subset of the data packets from the pacing send ring and the one or more of the another subset of the data packets from the bulk send ring for a predetermined transmit time, wherein the subset of the data packets are strictly prioritized over the another subset of the data packets and the one or more of the another subset of the data packets are only transmitted within the predetermined transmit time when the pacing send ring is emptied during the predetermined transmit time.

2. The method as set forth in claim **1**, further comprising repeating, by the network traffic management computing device, the releasing and transmitting for each other of the buckets.

3. The method as set forth in claim **1**, wherein the one or more of the subset of the data packets from the pacing send ring and the one or more of the another subset of the data packets from the bulk send ring are transmitted by a direct memory access (DMA) transmit engine.

4. The method as set forth in claim **1**, wherein the subset of the data packets to be transmitted is identified as paced and the another subset of the data packets is identified as bulk based on a data packet size, an associated type of traffic, or a quality of service (QoS) parameter.

5. The method as set forth in claim **1**, further comprising:

determining, by the network traffic management computing device, when another one of the buckets should be skipped based on a type of application from which the ones of the data packets associated with the selected data packet descriptors populated therein originated; and

skipping, by the network traffic management computing device, the another one of the buckets such that the releasing and transmitting are not repeated for the another one of the buckets, when the determining indicates that the another one of the buckets should be skipped.

6. The method as set forth in claim **1**, further comprising waiting, by the network traffic management computing device, to release another one of the buckets when the one or more of the subset of the data packets are transmitted from the pacing send ring and the one or more of the another subset of the data packets are transmitted from the bulk send ring prior to the expiration of the predetermined transmit time.

7. A non-transitory computer readable medium having stored thereon instructions for transmitting data packets at an optimized rate, comprising executable code which when executed by at least one processor and/or network interface causes the processor and/or network interface to perform steps comprising:

populating a plurality of buckets with one or more selected data packet descriptors associated with one or more corresponding ones of a subset of a plurality of data packets to be transmitted as paced and another subset of the data packets to be transmitted as bulk;

releasing one of the buckets to a hardware component comprising a pacing send ring and a bulk send ring, the releasing comprising writing one or more of the subset

of the data packets to the pacing send ring and one or more of the another subset of the data packets to the bulk send ring; and

transmitting the one or more of the subset of the data packets from the pacing send ring and the one or more of the another subset of the data packets from the bulk send ring for a predetermined transmit time, wherein the subset of the data packets are strictly prioritized over the another subset of the data packets and the one or more of the another subset of the data packets are only transmitted within the predetermined transmit time when the pacing send ring is emptied during the predetermined transmit time.

8. The non-transitory computer readable medium as set forth in claim **7**, wherein the executable code when executed by the processor and/or the network interface further causes the processor and/or the network interface to perform at least one additional step comprising repeating the releasing and transmitting for each other of the buckets.

9. The non-transitory computer readable medium as set forth in claim **7**, wherein the one or more of the subset of the data packets from the pacing send ring and the one or more of the another subset of the data packets from the bulk send ring are transmitted by a direct memory access (DMA) transmit engine.

10. The non-transitory computer readable medium as set forth in claim **7**, wherein the subset of the data packets to be transmitted is identified as paced and the another subset of the data packets is identified as bulk based on a data packet size, an associated type of traffic, or a quality of service (QoS) parameter.

11. The non-transitory computer readable medium as set forth in claim **7**, wherein the executable code when executed by the processor and/or the network interface further causes the processor and/or the network interface to perform at least one additional step comprising:

determining when another one of the buckets should be skipped based on a type of application from which the ones of the data packets associated with the selected data packet descriptors populated therein originated; and

skipping the another one of the buckets such that the releasing and transmitting are not repeated for the another one of the buckets, when the determining indicates that the another one of the buckets should be skipped.

12. The non-transitory computer readable medium as set forth in claim **7**, wherein the executable code when executed by the processor and/or the network interface further causes the processor and/or the network interface to perform at least one additional step comprising waiting to release another one of the buckets when the one or more of the subset of the data packets are transmitted from the pacing send ring and the one or more of the another subset of the data packets are transmitted from the bulk send ring prior to the expiration of the predetermined transmit time.

13. A network traffic management computing device comprising at least one processor and/or network interface and a memory coupled to the processor and/or network interface which is configured to be capable of executing programmed instructions comprising and stored in the memory to:

populate a plurality of buckets with one or more selected data packet descriptors associated with one or more corresponding ones of a subset of a plurality of data packets to be transmitted as paced and another subset of the data packets to be transmitted as bulk;

release one of the buckets to a hardware component comprising a pacing send ring and a bulk send ring, the releasing comprising writing one or more of the subset of the data packets to the pacing send ring and one or more of the another subset of the data packets to the bulk send ring; and

transmit the one or more of the subset of the data packets from the pacing send ring and the one or more of the another subset of the data packets from the bulk send ring for a predetermined transmit time, wherein the subset of the data packets are strictly prioritized over the another subset of the data packets and the one or more of the another subset of the data packets are only transmitted within the predetermined transmit time when the pacing send ring is emptied during the predetermined transmit time.

14. The network traffic management computing device as set forth in claim **13**, wherein the processor and/or network interface coupled to the memory is further configured to be capable of executing at least one additional programmed instruction comprising and stored in the memory to repeat the releasing and transmitting for each other of the buckets.

15. The network traffic management computing device as set forth in claim **13**, wherein the one or more of the subset of the data packets from the pacing send ring and the one or more of the another subset of the data packets from the bulk send ring are transmitted by a direct memory access (DMA) transmit engine.

16. The network traffic management computing device as set forth in claim **13**, wherein the subset of the data packets to be transmitted is identified as paced and the another subset of the data packets is identified as bulk based on a data packet size, an associated type of traffic, or a quality of service (QoS) parameter.

17. The network traffic management computing device as set forth in claim **13**, wherein the processor and/or network interface coupled to the memory is further configured to be capable of executing at least one additional programmed instruction comprising and stored in the memory to:

determine when another one of the buckets should be skipped based on a type of application from which the ones of the data packets associated with the selected data packet descriptors populated therein originated; and

skip the another one of the buckets such that the releasing and transmitting are not repeated for the another one of the buckets, when the determining indicates that the another one of the buckets should be skipped.

18. The network traffic management computing device as set forth in claim **13**, wherein the processor and/or network interface coupled to the memory is further configured to be capable of executing at least one additional programmed instruction comprising and stored in the memory to wait to release another one of the buckets when the one or more of the subset of the data packets are transmitted from the pacing send ring and the one or more of the another subset of the data packets are transmitted from the bulk send ring prior to the expiration of the predetermined transmit time.

* * * * *