(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 March 2020 (12.03.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/047740 A1

(51) International Patent Classification:
H04L 29/06 (2006.01)

(21) International Application Number:
PCT/CN2018/103967

(22) International Filing Date:
04 September 2018 (04.09.2018)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **ALIBABA GROUP HOLDING LIMITED** [—/CN]; Fourth Floor, One Capital Place, P.O. Box 847, George Town, Grand Cayman (KY).

(72) Inventors: **YU, Xiangning**; 14595 Oak St, Saratoga, California 95070 (US). **MA, Ke**; 22532 NE 98th Pl Redmond, Washington 98053 (US). **DUAN, Jianjun**; 6008 140th Ave SE Bellevue, Washington 98006 (US). **LIU, Kun**; Building 1, No. 969 West Wen Yi Road, Yu Hang District, Hangzhou, Zhejiang (CN).

(74) Agent: **BEIJING TSINGYUANHUI INTELLECTUAL PROPERTY LAW FIRM**; Room 362, 3rd Floor, Suzhou Street No. 1, Haidian District, Beijing 100080 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) Title: LOCKLESS PIPELINED NETWORK DATA PACKET BANDWIDTH CONTROL

(57) Abstract: Systems and methods are provided for improving multi-threading performance in a computing system with multiple processors by reducing latency and controlling bandwidth by grouping network data packets associated with processors into bandwidth groups based on a bandwidth requirement, aggregating the network data packets associated with processors into a lockless drain queue, performing bandwidth control on the lockless drain queue based on a respective length of each network data packet, and sending an inter-processor interrupt (IPI) to the corresponding processor of a currently queued network data packet in the lockless drain queue to continue processing the currently queued network data packet.

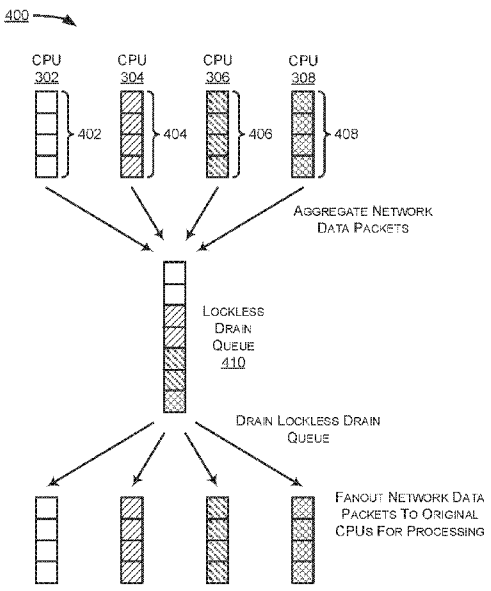

FIG. 4

WO 2020/047740 A1

LOCKLESS PIPELINED NETWORK DATA PACKET BANDWIDTH CONTROL

BACKGROUND

[0001] In recent years, many businesses have increased utilizing computing service providers to manage data storage, computation, and the like. These computing service providers support varying service needs for different businesses, which may be referred to as a mixed, or a multi-tenancy, scenario. In a multi-tenancy scenario for a computing service provider, different services share the same physical network interface card (NIC), however, different services have very different requirements for network service quality. For example, online businesses, such as e-commerce, may require low latency and high packets per second (PPS), while offline services, such as big data, may not be sensitive to latency but may have high bandwidth requirements. Therefore, a flow, or traffic group, which is used to isolate and share network resources, needs to be managed effectively to meet different demands and requirements through the same NIC.

[0002] Currently, some commercially available traffic control (TC) technical solutions require a very high percentage, up to 100%, occupation of some central processing unit (CPU) resources of a server to manage the flow control without adequately meeting the high PPS requirement, or are not suitable for a large-scale deployment. For example, some flow control algorithms allocate some CPUs in a system to run the flow control algorithm in a manner of a busy loop,

which consumes valuable CPU resources without ensuring the network data packet to be processed by the original CPU. If the network data packet were not processed by the original CPU, the performance of the system would degrade due to a problem with cache locality of the system. Other control algorithms involve simultaneous access to current bandwidth usage from different threads among a large number of CPUs, and implement locks for both bandwidth sharing and bandwidth limitation. Performance of a system utilizing such a control algorithm is usually limited by global locks, which results in a degradation of multithreading performance.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] The detailed description is set forth with reference to the accompanying figures. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The use of the same reference numbers in different figures indicates similar or identical items or features.

[0004] FIG. 1 illustrates an example process of a bandwidth based network data packet control.

[0005] FIG. 2 illustrates an example flowchart detailing one of blocks of FIG. 1.

[0006] FIG. 3 illustrates an example system for implementing the processes and methods described above for a bandwidth based network data packet control.

[0007] FIG. 4 illustrates a pictorial representation 400 of the bandwidth based network data packet control.

DETAILED DESCRIPTION

[0008] Methods and systems discussed herein are directed to improving network resource isolation, and more specifically to improving multithreading performance in a computing system with multiple processors by reducing latency and controlling bandwidth.

[0009] In a multi-tenancy scenario, a system shares the same physical network interface card (NIC) for different services that have different requirements for network service quality. The system may include a plurality of processors, such as CPUs, for controlling network data packets associated with the plurality of CPUs. The CPUs may include physical CPUs running on physical machines and virtual CPUs running on the virtual machines. Instead of having a CPU dedicated for flow, or traffic group, each CPU may be scheduled to perform the flow control task based on a predetermined schedule, such as a round-robin manner providing certain performance fairness among the CPUs, i.e., no single CPU is to perform the flow control task more than other CPUs. The currently scheduled CPU may also be referred to as an executing CPU. Each CPU may hold its network data packets in its own lockless queue. The executing CPU may group network data packets associated with the plurality of CPUs into bandwidth groups based on a corresponding bandwidth requirement and aggregate the

network data packets into a lockless drain queue. As an aggregation thread, the executing CPU may aggregate the network data packets into the lockless drain queue by scanning a queue of each CPU of the plurality of CPUs at a high rate, for example, scanning every one to two microseconds, and queueing the respective network data packets associated with the plurality of CPUs in a predetermined order of the plurality of CPUs, such as a round-robin manner. In this aggregation phase, no contents of the network data packets are accessed. The network data packets are, therefore, likely to remain in the cache of the corresponding original CPUs ensuring the cache locality of the network data packets and avoiding problems of data being simultaneously accessed by multiple threads.

[0010] The executing CPU may then perform bandwidth control on the lockless drain queue based on a respective length of each network data packet, for example by performing a token bucket algorithm on the lockless drain queue or using a bandwidth group dedicated thread, and, as a drain thread, send an inter-processor interrupt (IPI) to the corresponding CPU of a currently queued network data packet in the lockless drain queue to continue processing the queued network data packet.

[0011] The system, by executing the process described above, may effectively align the aggregation thread and the drain thread with the bandwidth group, and align the enqueue and dequeue threads with appropriate CPUs such that the original sending CPUs retain and process their network data packets, which allows better concurrency for different process, or pipeline, stages.

[0012] To further improve the bandwidth performance of the system by decoupling bandwidth sharing and bandwidth limitation, the executing CPU may run a bandwidth allocation thread on one of the plurality of CPUs at a low frequency, such as a rate of once per second, and dynamically adjust a corresponding maximum bandwidth of each of the bandwidth groups.

[0013] FIG. 1 illustrates an example process 100 of a bandwidth based network data packet control.

[0014] In a system with a plurality of processors, such as CPUs, for controlling network data packets associated with the plurality of CPUs, each CPU may be scheduled at block 102 based on a predetermined schedule, and a currently scheduled CPU may be selected to perform or execute the network data packet controlling task during the scheduled interval at block 104. The CPUs may include physical CPUs running on physical machines and virtual CPUs running on virtual machines. The predetermined schedule may be generated in a round-robin manner, which may ensure certain fairness among the CPUs, that is, no single CPU may be scheduled to perform the network data packet controlling task more than other CPUs. Instead of having a separate dedicated CPU to perform the network data packet controlling task, the overhead cost, in material and complexity, may be reduced by spreading the network data packet controlling task among the existing network data packet processing CPUs. Each CPU may hold its network data packets in its own lockless queue.

[0015] At block 106, the executing CPU may group network data packets associated with the plurality of CPUs into bandwidth groups based on a

bandwidth requirement of a corresponding network data packet, and aggregate the network data packets into a lockless drain queue at block 108 as an aggregation thread. Because no contents of the network data packets may be accessed in this aggregation phase, the network data packets are likely to remain in the cache of the corresponding original CPUs, which may ensure the cache locality of the network data packets and avoid problems of data being simultaneously accessed by multiple threads.

[0016] At block 110, the executing CPU may perform bandwidth control on the lockless drain queue based on a length of each network data packet. For example, the executing CPU may perform a token bucket algorithm on the lockless drain queue or use a bandwidth group dedicated thread. As a drain thread, the executing CPU may then send an inter-processor interrupt (IPI) to a CPU of a currently queued network data packet in the lockless drain queue to continue processing the currently queued network data packet at block 112.

[0017] The process described above with reference to FIG. 1, may effectively align the aggregation thread and the drain thread with the bandwidth group, and improve concurrency for different process, or pipeline, stages by aligning the enqueue and dequeue threads with appropriate CPUs such that the original sending CPUs retain and process their network data packets.

[0018] FIG. 2 illustrates an example flowchart detailing block 108 of FIG. 1.

[0019] At block 202, the executing CPU may scan a queue of each CPU of the plurality of CPUs at a high frequency, for example, at a rate of between once per one microsecond and once per two microseconds. The executing CPU may,

at block 204, queue the respective network data packets of the plurality of CPUs in a predetermined order of the plurality of CPUs in the lockless drain queue.

[0020] To improve bandwidth performance of the system by decoupling bandwidth sharing and bandwidth limitation, the executing CPU may simultaneously run a bandwidth allocation thread on one of the plurality of CPUs, and dynamically adjust a corresponding maximum bandwidth of each of the bandwidth groups. The executing CPU may run the bandwidth allocation thread on each CPU at a low frequency, for example, at a rate of once per second.

[0021] FIG. 3 illustrates an example system 300 for implementing the processes and methods described above for a bandwidth based network data packet control.

[0022] The techniques and mechanisms described herein may be implemented by multiple instances of the system 300 as well as by any other computing device, system, and/or environment including cloud computing. The system 300 shown in FIG. 3 is only one example of a system and is not intended to suggest any limitation as to the scope of use or functionality of any computing device utilized to perform the processes and/or procedures described above. Other well-known computing devices, systems, environments and/or configurations that may be suitable for use with the embodiments include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, game consoles, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that

include any of the above systems or devices, implementations using field programmable gate arrays ("FPGAs") and application specific integrated circuits ("ASICs"), and/or the like.

[0023] The system 300 may comprise a plurality of processors (four processors 302, 304, 306, and 308 are shown in this example as CPUs), and memory 310 coupled to the processors. Each of the processors 302, 304, 306, and 308 may, in turn, execute computer executable instructions stored in the memory 310, as an executing processor, to perform a variety of functions as described below with reference to FIGs. 1 and 2. The processors 302, 304, 306, and 308 may include a central processing unit (CPU), a graphics processing unit (GPU), both CPU and GPU, or other processing units or components known in the art. Additionally, each of the processors 302, 304, 306, and 308 may possess its own local memory, which also may store program modules, program data, and/or one or more operating systems.

[0024] Depending on the exact configuration and type of the system 300, the memory 310 may be volatile, such as RAM, non-volatile, such as ROM, flash memory, miniature hard drive, memory card, and the like, or some combination thereof. The memory 310 may include one or more computer-executable modules (two modules 312 and 314 are shown in this example) that are executable by the processors 302, 304, 306, and 308.

[0025] The system 300 may additionally include an input/output (I/O) interface 316 for receiving data, such as network data packets, and for outputting processed data. The system 300 may also include a communication module 318

and a network interface module 320 allowing the system 300 to communicate with other device(s) or system(s) 322 over a network 324. The network 324 may include the Internet, wired media such as a wired network or direct-wired connections, and wireless media such as acoustic, radio frequency (RF), infrared, and other wireless media.

[0026] In the system 300, an executing CPU may be selected based on a predetermined schedule, which may be generated in a round-robin manner, which may ensure certain fairness among the CPUs, that is, no single CPU may be scheduled to perform the network data packet controlling task more than other CPUs. For example, the order of serving as the executing CPU may be CPU 302, CPU 304, CPU 306, CPU 308, then back to CPU 302, and so on. An interval of serving as the executing CPU for each CPU may not be the same every time depending on processes being performed, however, over time, it will average out. Instead of having a separate dedicated CPU to perform the network data packet controlling task, the overhead cost, in material and complexity, may be reduced by spreading the network data packet controlling task among the existing network data packet processing CPUs. Each CPU may hold its network data packets in its own lockless queue.

[0027] The executing CPU, the CPU 302 in this example, may group network data packets associated with the CPUs 302, 304, 306, and 308 into bandwidth groups based on a bandwidth requirement of a corresponding network data packet, and aggregate the network data packets associated with the CPUs 302, 304, 306, ad 308 into a lockless drain queue as an aggregation thread. The

executing CPU 302 may scan a queue of each of CPUs 302, 304, 306, and 308, at a high frequency, for example at a rate of once between one to two microseconds, and queue the respective network data packets associated with the CPUs 302, 304, 306, and 308 in the lockless drain queue in a predetermined order of the CPUs 302, 304, 306, and 308, which may be based on a round-robin manner. In this aggregation phase, no contents of the network data packets are accessed. The network data packets are, therefore, likely to remain in the cache of the corresponding original CPUs ensuring the cache locality of the network data packets and avoiding problems caused by mutex, i.e., data being simultaneously accessed by multiple threads.

[0028] The executing CPU 302 may then perform bandwidth control on the lockless drain queue based on a respective length of each network data packet, and, as a drain thread, send an inter-processor interrupt (IPI) to the CPU of a currently queued network data packet in the lockless drain queue to continue processing the currently queued network data packet. The executing CPU 302 may perform the bandwidth control on the lockless drain queue by performing a token bucket algorithm on the lockless drain queue or using a bandwidth group dedicated thread.

[0029] The system described above with reference to FIG. 3, may effectively align the aggregation thread and the drain thread with the bandwidth group, and improve concurrency for different process, or pipeline, stages by aligning the enqueue and dequeue threads with appropriate CPUs such that the original sending CPUs retain and process their network data packets.

[0030] To improve bandwidth performance of the system 300 by decoupling bandwidth sharing and bandwidth limitation, the executing CPU 302 may run a bandwidth allocation thread on one of the plurality of CPUs at a low frequency, for example, once a second, and dynamically adjust a corresponding maximum bandwidth of each of the bandwidth groups to a maximum available bandwidth.

[0031] FIG. 4 illustrates a pictorial representation 400 of the bandwidth based network data packet control.

[0032] The network data packets, 402, 404, 406, and 408 may be held in lockless queues of the CPUs 302, 304, 306, and 308, respectively. The executing CPU 302 may group the network data packets 402, 404, 406, and 408 into bandwidth groups based on each network data packets in the network data packets 402, 404, 406, and 408 based on the bandwidth requirement of each network data packet, and aggregate the network data packets into the lockless drain queue 410 in a predetermined order of the CPUs as described above with reference to FIG. 1, 2, and 3. The executing CPU 302 may then perform bandwidth control on the lockless drain queue 410 based on a length of each network data packet, for example, by performing a token bucket algorithm on the lockless drain queue 410. The executing CPU 302 may then send an inter-processor interrupt (IPI) to a CPU, such as the CPU 308, of a currently queued network data packet (a network data packet from the network data packets 408 of the CPU 308 shown) in the lockless drain queue 430 to continue processing the currently queued network data packet.

[0033] Some or all operations of the methods described above can be performed by execution of computer-readable instructions stored on a computer-readable storage medium, as defined below. The term "computer-readable instructions" as used in the description and claims, include routines, applications, application modules, program modules, programs, components, data structures, algorithms, and the like. Computer-readable instructions can be implemented on various system configurations, including single-processor or multiprocessor systems, minicomputers, mainframe computers, personal computers, hand-held computing devices, microprocessor-based, programmable consumer electronics, combinations thereof, and the like.

[0034] The computer-readable storage media may include volatile memory (such as random access memory (RAM)) and/or non-volatile memory (such as read-only memory (ROM), flash memory, etc.). The computer-readable storage media may also include additional removable storage and/or non-removable storage including, but not limited to, flash memory, magnetic storage, optical storage, and/or tape storage that may provide non-volatile storage of computer-readable instructions, data structures, program modules, and the like.

[0035] A non-transient computer-readable storage medium is an example of computer-readable media. Computer-readable media includes at least two types of computer-readable media, namely computer-readable storage media and communications media. Computer-readable storage media includes volatile and non-volatile, removable and non-removable media implemented in any process or technology for storage of information such as computer-readable instructions,

data structures, program modules, or other data. Computer-readable storage media includes, but is not limited to, phase change memory (PRAM), static random-access memory (SRAM), dynamic random-access memory (DRAM), other types of random-access memory (RAM), read-only memory (ROM), electrically erasable programmable read-only memory (EEPROM), flash memory or other memory technology, compact disk read-only memory (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 non-transmission medium that can be used to store information for access by a computing device. The memory 310 is an example of the computer-readable storage media. In contrast, communication media may embody computer-readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave, or other transmission mechanism. As defined herein, computer-readable storage media do not include communication media.

[0036] The computer-readable instructions stored on one or more non-transitory computer-readable storage media that, when executed by one or more processors, may perform operations described above with reference to FIGs. 1-4. Generally, computer-readable instructions include routines, programs, objects, components, data structures, and the like that perform particular functions or implement particular abstract data types. The order in which the operations are described is not intended to be construed as a limitation, and any number of the

described operations can be combined in any order and/or in parallel to implement the processes.

EXAMPLE CLAUSES

[0037] A. A method in a system comprising a plurality of central processing units (CPUs) for controlling network data packets associated with the plurality of CPUs, the method comprising: grouping network data packets associated with the plurality of CPUs into bandwidth groups based on a bandwidth requirement of a corresponding network data packet; aggregating the network data packets associated with the plurality of CPUs into a lockless drain queue; and performing bandwidth control on the lockless drain queue based on a respective length of each network data packet.

[0038] B. The method as paragraph A recites, further comprising sending an inter-processor interrupt (IPI) to a corresponding CPU of a currently queued network data packet in the lockless drain queue to continue processing the currently queued network data packet.

[0039] C. The method as paragraph A recites, prior to grouping the network data packets associated with f the plurality of CPUs into bandwidth groups based on the bandwidth requirement, further comprising: scheduling each CPU of the plurality of CPUs based on a predetermined schedule; and selecting a currently scheduled CPU, wherein the method of paragraph A is performed by the currently scheduled CPU.

[0040] D. The method as paragraph A recites, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes scanning a queue of each CPU of the plurality of CPUs at a rate of between once per one microsecond and once per two microseconds.

[0041] E. The method as paragraph A recites, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes: queuing the respective network data packets associated with the plurality of CPUs in a predetermined order of the plurality of CPUs in the lockless drain queue.

[0042] F. The method as paragraph A recites, wherein performing the bandwidth control on the lockless drain queue based on the respective length of each network data packet includes using a bandwidth group dedicated thread.

[0043] G. The method as paragraph A recites, further comprising: running a bandwidth allocation thread on one of the plurality of CPUs; and dynamically adjusting a corresponding maximum bandwidth of each of the bandwidth groups.

[0044] H. The method as paragraph F recites, wherein running the bandwidth allocation thread on the one CPU includes running the bandwidth allocation thread on the one CPU at a rate of once per second.

[0045] I. The method as paragraph A recites, wherein network data packets are held in a lockless queue of a corresponding CPU.

[0046] J. A system comprising: a plurality of central processing units (CPUs); memory coupled to the plurality of CPUs, the memory storing computer

executable instructions executable by any one of the plurality of CPUs, that when executed, causes an executing CPU of the plurality of CPUs to perform operations comprising: grouping network data packets associated with the plurality of CPUs into bandwidth groups based on a bandwidth requirement of a corresponding network data packet; aggregating the network data packets associated with the plurality of CPUs into a lockless drain queue; and performing bandwidth control on the lockless drain queue based on a respective length of each network data packet.

[0047] K. The system as paragraph J recites, wherein the operations further comprise sending an inter-processor interrupt (IPI) to a corresponding CPU of a currently queued network data packet in the lockless drain queue to continue processing the currently queued network data packet.

[0048] L. The system as paragraph J recites, wherein the executing CPU is selected based on a predetermined schedule.

[0049] M. The system as paragraph J recites, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes: scanning a queue of each CPU of the plurality of CPUs at a rate of between once per one microsecond and once per two microseconds.

[0050] N. The system as paragraph J recites, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes: queuing the respective network data packets associated with the plurality of CPUs in a predetermined order of the plurality of CPUs in the lockless drain queue.

[0051] O. The system as paragraph J recites, wherein performing the bandwidth control on the lockless drain queue based on the respective length of each network data packet includes using a bandwidth group dedicated thread.

[0052] P. The system as paragraph J recites, wherein the operations further comprise: running a bandwidth allocation thread on one of the plurality of CPUs, and dynamically adjusting a corresponding maximum bandwidth of each of the bandwidth groups.

[0053] Q. The system as paragraph P recites, wherein running the bandwidth allocation thread on the one CPU includes running the bandwidth allocation thread on the one CPU at a rate of once per second.

[0054] R. The system as paragraph J recites: wherein network data packets are held in a lockless queue of a corresponding CPU.

[0055] S. A computer readable medium storing computer-readable instructions executable by any one of a plurality of CPUs in a system, that when executed, causes an executing CPU of the plurality of CPUs to perform operations comprising: grouping network data packets associated with the plurality of CPUs into bandwidth groups based on a bandwidth requirement of a corresponding network data packet; aggregating the network data packets associated with the plurality of CPUs into a lockless drain queue; and performing bandwidth control on the lockless drain queue based on a respective length of each network data packet.

[0056] T. The computer readable medium as paragraph S recites, wherein the operations further comprise sending an inter-processor interrupt (IPI)

to a corresponding CPU of a currently queued network data packet in the lockless drain queue to continue processing the currently queued network data packet.

[0057] U. The computer readable medium as paragraph S recites, wherein the executing CPU is selected based on a predetermined schedule.

[0058] V. The computer readable medium as paragraph S recites, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes: scanning a queue of each CPU of the plurality of CPUs at a rate of between once per one microsecond and once per two microseconds.

[0059] W. The computer readable medium as paragraph S recites, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes: queuing the respective network data packets associated with the plurality of CPUs in a predetermined order of the plurality of CPUs in the lockless drain queue.

[0060] X. The computer readable medium as paragraph S recites, wherein performing the bandwidth control on the lockless drain queue based on the respective length of each network data packet includes using a bandwidth group dedicated thread.

[0061] Y. The computer readable medium as paragraph S recites, wherein the operations further comprise: running a bandwidth allocation thread on one of the plurality of CPUs, and dynamically adjusting a corresponding maximum bandwidth of each of the bandwidth groups.

[0062] Z. The computer readable medium as paragraph Y recites, wherein running the bandwidth allocation thread on the one CPU includes running the bandwidth allocation thread on the one CPU at a rate of once per second.

[0063] AA. The computer readable medium as paragraph S recites, wherein network data packets are held in a lockless queue of a corresponding CPU.

CONCLUSION

[0064] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the claims.

CLAIMS

WHAT IS CLAIMED IS:

1. A method in a system comprising a plurality of central processing units (CPUs) for controlling network data packets associated with the plurality of CPUs, the method comprising:

grouping network data packets associated with the plurality of CPUs into bandwidth groups based on a bandwidth requirement of a corresponding network data packet;

aggregating the network data packets associated with the plurality of CPUs into a lockless drain queue; and

performing bandwidth control on the lockless drain queue based on a respective length of each network data packet.

2. The method of claim 1, further comprising:

sending an inter-processor interrupt (IPI) to a corresponding CPU of a currently queued network data packet in the lockless drain queue to continue processing the currently queued network data packet.

3. The method of claim 1, wherein prior to grouping the network data packets associated with the plurality of CPUs into bandwidth groups based on the bandwidth requirement, the method further comprises:

scheduling each CPU of the plurality of CPUs based on a predetermined schedule; and

selecting a currently scheduled CPU,

wherein the method of claim 1 is performed by the currently scheduled CPU.

4. The method of claim 1, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes:

scanning a queue of each CPU of the plurality of CPUs at a rate of between once per one microsecond and once per two microseconds.

5. The method of claim 1, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes:

queuing the respective network data packets associated with the plurality of CPUs in a predetermined order of the plurality of CPUs in the lockless drain queue.

6. The method of claim 1, wherein performing the bandwidth control on the lockless drain queue based on the respective length of each network data packet includes using a bandwidth group dedicated thread.

7. The method of claim 1, further comprising:

running a bandwidth allocation thread on one CPU of the plurality of CPUs; and

dynamically adjusting a corresponding maximum bandwidth of each of the bandwidth groups.

8. The method of claim 7, wherein running the bandwidth allocation thread on the one CPU of the plurality of CPUs includes running the bandwidth allocation thread on the one CPU of the plurality of CPUs at a rate of once per second.

9. The method of claim 1, wherein network data packets are held in a lockless queue of a corresponding CPU.

10. A system comprising:

a plurality of central processing units (CPUs);

memory coupled to the plurality of CPUs, the memory storing computer executable instructions executable by any one of the plurality of CPUs, that when executed, causes an executing CPU of the plurality of CPUs to perform operations comprising:

grouping network data packets associated with the plurality of CPUs into bandwidth groups based on a bandwidth requirement of a corresponding network data packet;

aggregating the network data packets associated with the plurality of CPUs into a lockless drain queue; and

performing bandwidth control on the lockless drain queue based on a respective length of each network data packet.

11. The system of claim 10, wherein the operations further comprise:

sending an inter-processor interrupt (IPI) to a corresponding CPU of a currently queued network data packet in the lockless drain queue to continue processing the currently queued network data packet.

12. The system of claim 11, wherein the executing CPU is selected based on a predetermined schedule.

13. The system of claim 11, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes:

scanning a queue of each CPU of the plurality of CPUs at a rate of between once per one microsecond and once per two microseconds.

14. The system of claim 11, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes:

queuing the respective network data packets associated with the plurality of CPUs in a predetermined order of the plurality of CPUs in the lockless drain queue.

15. The system of claim 11, wherein performing the bandwidth control on the lockless drain queue based on the respective length of each network data packet includes using a bandwidth group dedicated thread.

16. The system of claim 11, wherein the operations further comprise:

running a bandwidth allocation thread on one CPU of the plurality of CPUs, and

dynamically adjusting a corresponding maximum bandwidth of each of the bandwidth groups.

17. The system of claim 16, wherein running the bandwidth allocation thread on the one CPU of the plurality of CPUs includes running the bandwidth allocation thread on the one CPU of the plurality of CPUs at a rate of once per second.

18. The system of claim 11, wherein network data packets are held in a lockless queue of a corresponding CPU.

19. A computer readable medium storing computer-readable instructions executable by any one of a plurality of CPUs in a system, that when executed, causes an executing CPU of the plurality of CPUs to perform operations comprising:

grouping network data packets associated with the plurality of CPUs into bandwidth groups based on a bandwidth requirement of a corresponding network data packet;

aggregating the network data packets associated with the plurality of CPUs into a lockless drain queue; and

performing bandwidth control on the lockless drain queue based on a respective length of each network data packet.

20. The computer readable medium of claim 19, wherein the operations further comprise:

sending an inter-processor interrupt (IPI) to a corresponding CPU of a currently queued network data packet in the lockless drain queue to continue processing the currently queued network data packet.

21. The computer readable medium of claim 20, wherein the executing CPU is selected based on a predetermined schedule.

22. The computer readable medium of claim 20, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes:

scanning a queue of each CPU of the plurality of CPUs at a rate of between once per one microsecond and once per two microseconds.

23. The computer readable medium of claim 20, wherein aggregating the network data packets associated with the plurality of CPUs into the lockless drain queue includes:

queuing the respective network data packets associated with the plurality of CPUs in a predetermined order of the plurality of CPUs in the lockless drain queue.

24. The computer readable medium of claim 20, wherein performing the bandwidth control on the lockless drain queue based on the respective length of each network data packet includes using a bandwidth group dedicated thread.

25. The computer readable medium of claim 20, wherein the operations further comprise:

running a bandwidth allocation thread on one CPU of the plurality of CPUs, and

dynamically adjusting a corresponding maximum bandwidth of each of the bandwidth groups.

26. The computer readable medium of claim 25, wherein running the bandwidth allocation thread on the one CPU of the plurality of CPUs includes running the bandwidth allocation thread on the one CPU of the plurality of CPUs at a rate of once per second.

27. The computer readable medium of claim 20, wherein network data packets are held in a lockless queue of a corresponding CPU.

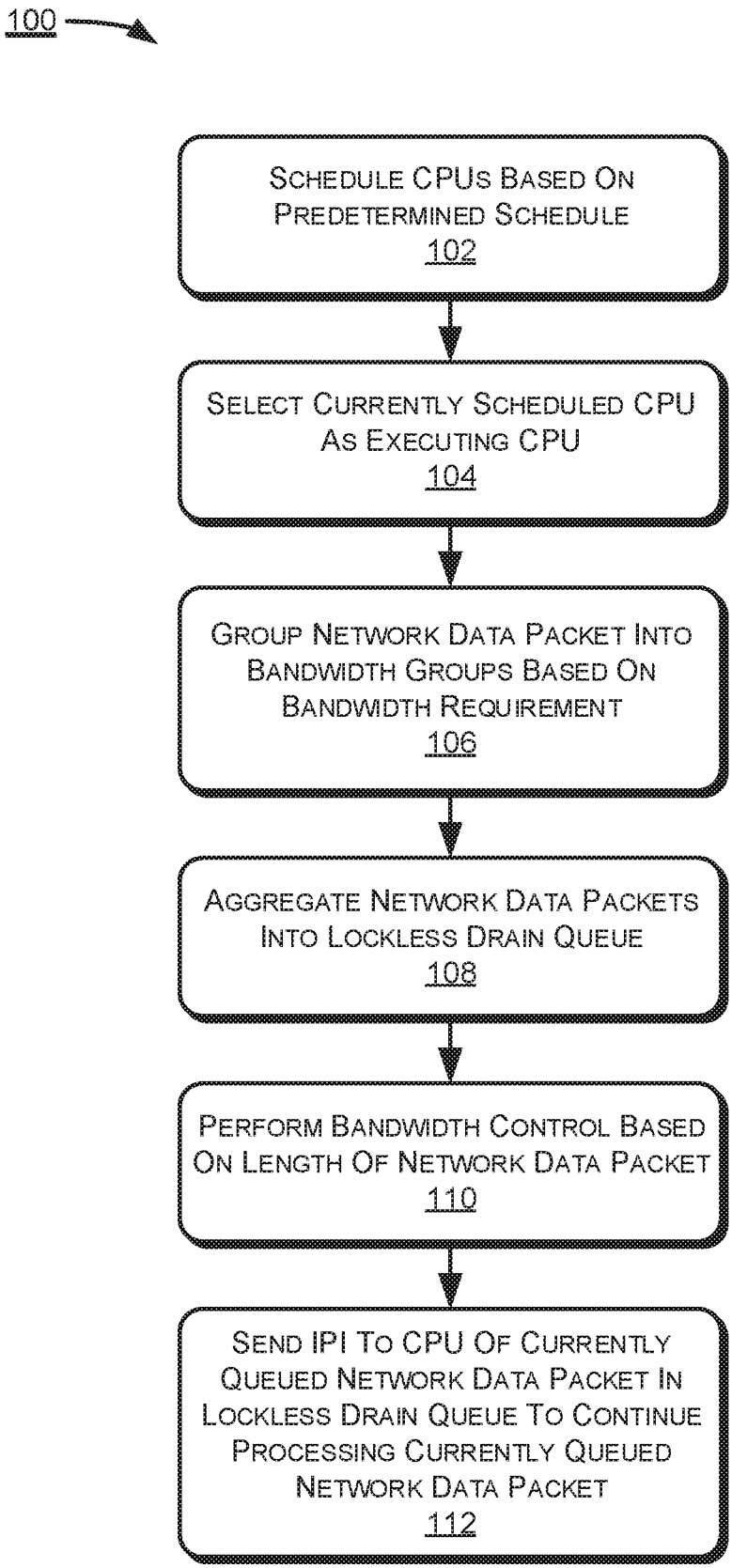
100
SCHEDULE CPUS BASED ON PREDETERMINED SCHEDULE
102
SELECT CURRENTLY SCHEDULED CPU AS EXECUTING CPU
104
GROUP NETWORK DATA PACKET INTO BANDWIDTH GROUPS BASED ON BANDWIDTH REQUIREMENT
106
AGGREGATE NETWORK DATA PACKETS INTO LOCKLESS DRAIN QUEUE
108
PERFORM BANDWIDTH CONTROL BASED ON LENGTH OF NETWORK DATA PACKET
110
SEND IPI TO CPU OF CURRENTLY QUEUED NETWORK DATA PACKET IN LOCKLESS DRAIN QUEUE TO CONTINUE PROCESSING CURRENTLY QUEUED NETWORK DATA PACKET
112

FIG. 1

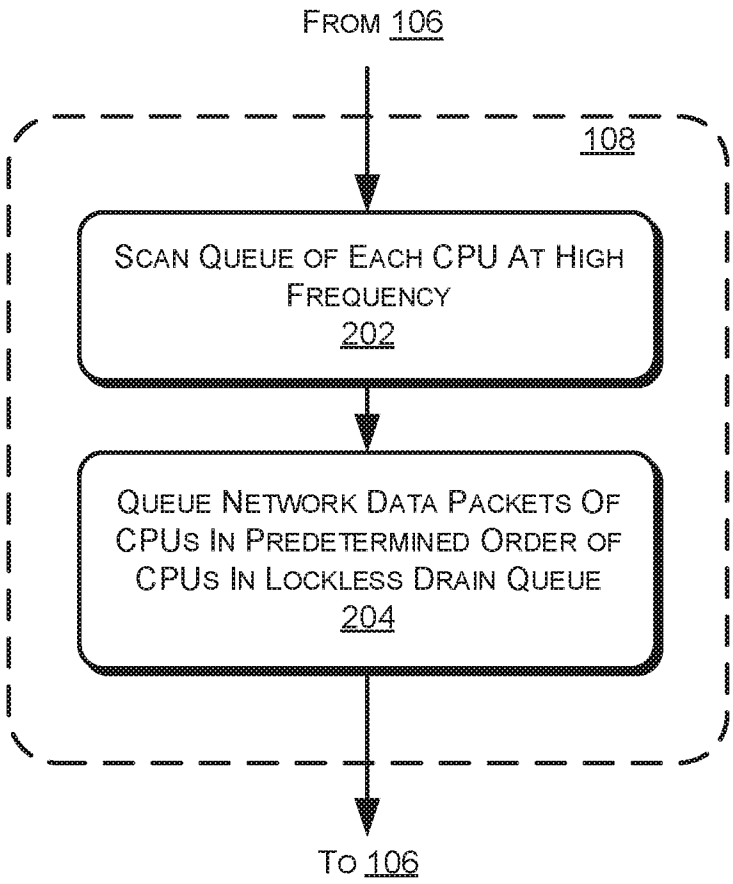
FROM 106
108
SCAN QUEUE OF EACH CPU AT HIGH FREQUENCY
202
QUEUE NETWORK DATA PACKETS OF CPUS IN PREDETERMINED ORDER OF CPUS IN LOCKLESS DRAIN QUEUE
204
TO 106

FIG. 2

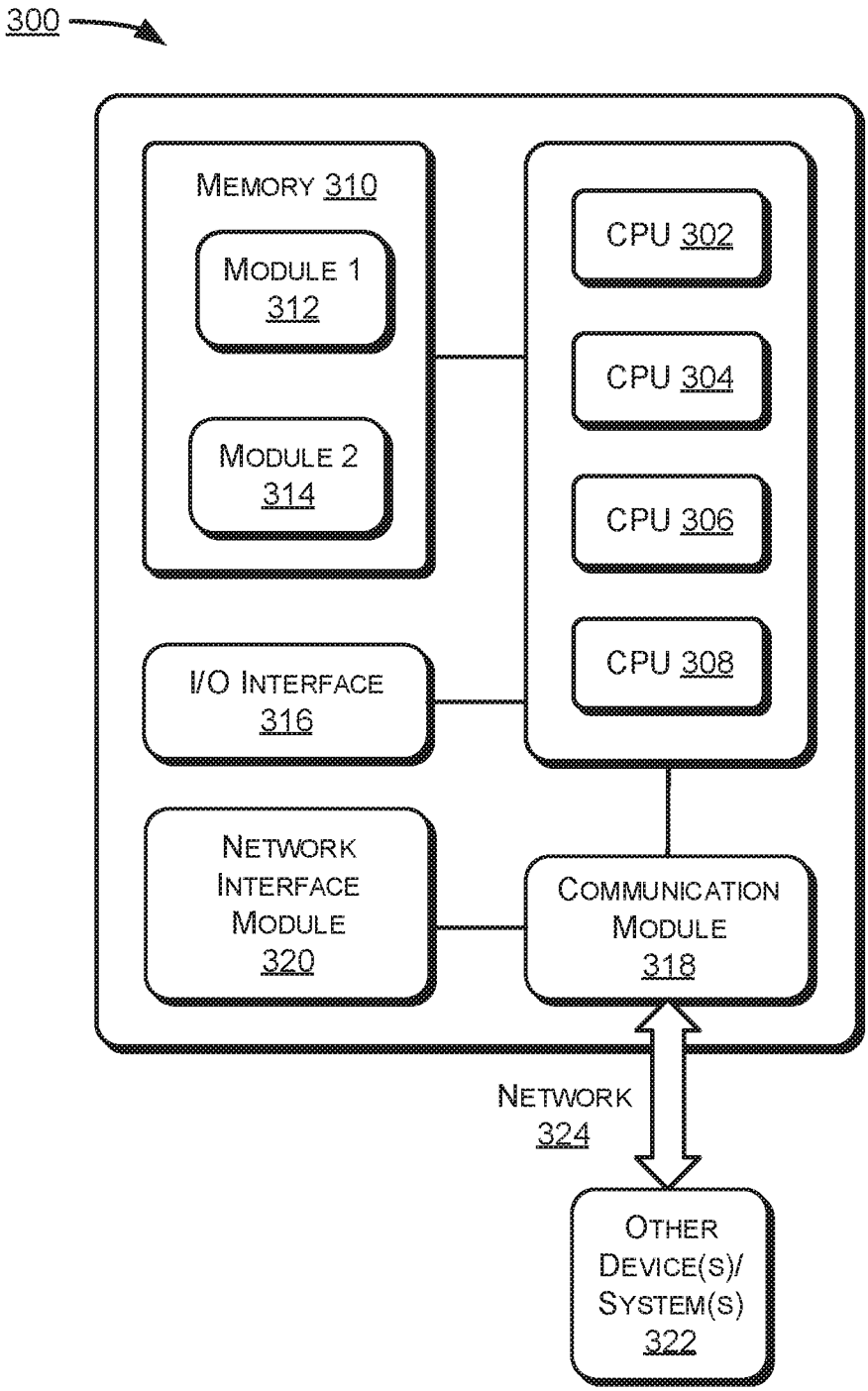
300
MEMORY 310
MODULE 1 312
MODULE 2 314
CPU 302
CPU 304
CPU 306
CPU 308
I/O INTERFACE 316
NETWORK INTERFACE MODULE 320
COMMUNICATION MODULE 318
NETWORK 324
OTHER DEVICE(S)/ SYSTEM(S) 322

FIG. 3

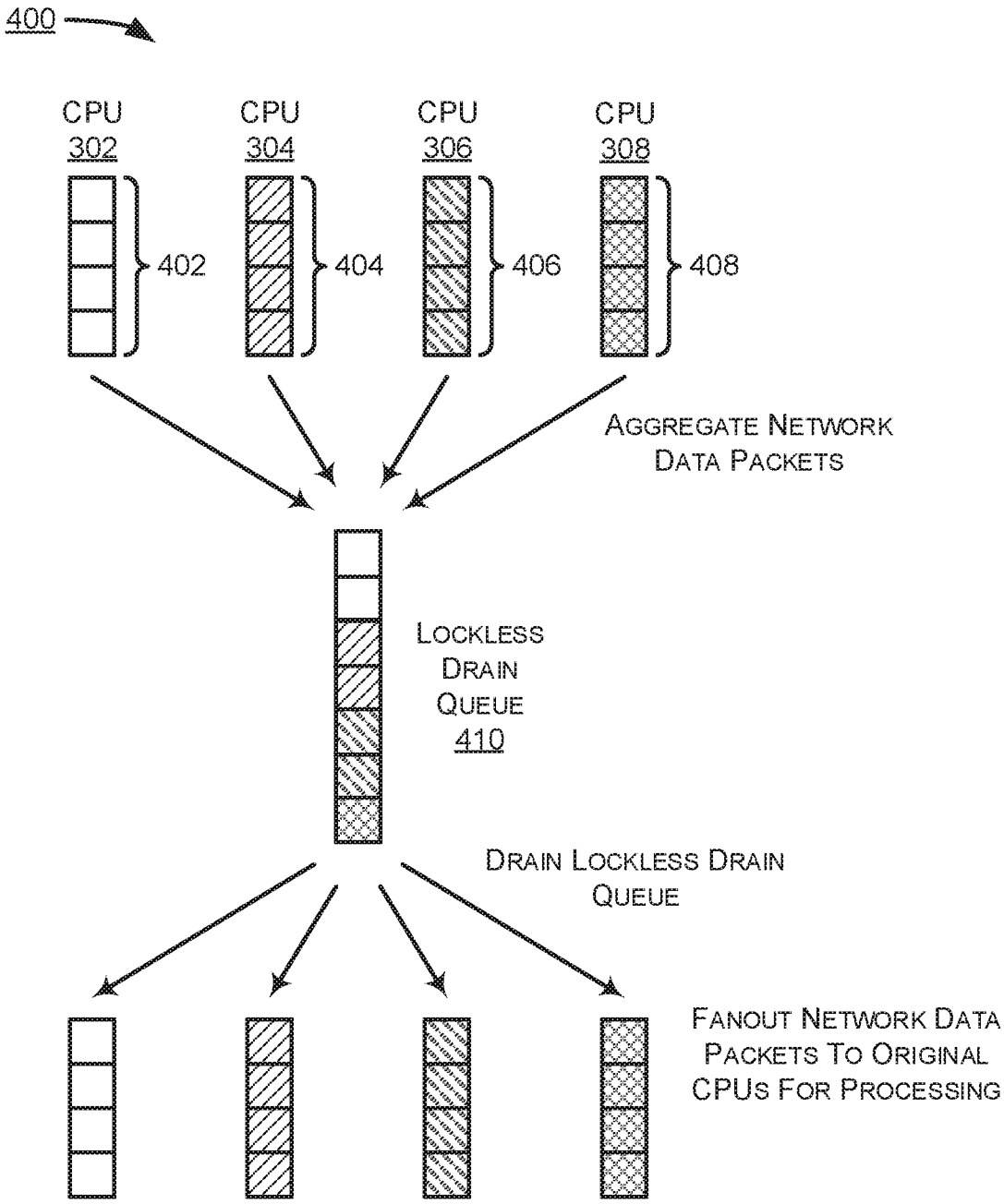
400
CPU 302
CPU 304
CPU 306
CPU 308
402
404
406
408
AGGREGATE NETWORK DATA PACKETS
LOCKLESS DRAIN QUEUE 410
DRAIN LOCKLESS DRAIN QUEUE
FANOUT NETWORK DATA PACKETS TO ORIGINAL CPUS FOR PROCESSING

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2018/103967

A. CLASSIFICATION OF SUBJECT MATTER

H04L 29/06(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L; G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNABS,CNTXT,CNKI,VEN,USTXT: multi, plurality, CPU, processor, core, thread, packet, package, bandwidth, queue, length

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2006187945 A1 (BROADCOM CORP) 24 August 2006 (2006-08-24) the whole document	1-27
A	US 2006227788 A1 (ELDAR AVIGDORET AL.) 12 October 2006 (2006-10-12) the whole document	1-27
A	CN 103957470 A (ZHEJIANG UNIVERSITY OF WATER RESOURCES AND ELECTRIC POWER ET AL.) 30 July 2014 (2014-07-30) the whole document	1-27

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 May 2019	**21 May 2019**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing** **100088** **China**	Authorized officer **YUE,Jin**
Facsimile No. **(86-10)62019451**	Telephone No. **86-(010)-62088427**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2018/103967

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2006187945 A1	24 August 2006	US 7948896 B2	24 May 2011
US 2006227788 A1	12 October 2006	None	
CN 103957470 A	30 July 2014	None	