

CONTROL PLANE STRUCTURE FOR COMMUNICATING BETWEEN A HOST AND A SHARED NETWORK ADAPTER

BACKGROUND

[0001] The present invention relates to a control plane structure for establishing communications between a host and a shared network adapter.

[0002] A shared resource environment enables workloads executing within the environment, even those of different customers, to be consolidated on one machine allowing the resources of that machine to be shared.

[0003] One example of a shared resource is a shared network interface (e.g., a shared adapter), which facilitates communications with one or more hosts coupled to the shared network interface. The shared network interface facilitates the transfer of data, including large amounts of data, into or out of a host and the host's file system. It also facilitates the streaming of other types of large data, such as video or complex engineering or scientific graphics. Transferring large amounts of data, whether it be files, streams, or other data, can be very expensive in terms of central processing unit (CPU) cycles and the cost of the network input/output (I/O).

SUMMARY

[0004] According to one embodiment of the present invention, a method includes activating control queues between a host and a shared network adapter, issuing control commands to the shared network adapter using the control queues where the control commands configure a data plane for exchanging data between the host and the shared network adapter, activating data queues between the host and the shared network adapter after issuing the control commands, and upon identifying an error, suspending the data queues while permitting the control queues to still transmit control plane data between the host and the shared network adapter.

[0005] According to one embodiment of the present invention is a system that includes one or more processors configured to host a plurality of logical partitions or virtual machines, a shared network adapter configured to provide an interface between the plurality of logical partitions or virtual machines and a network interface card/controller (NIC), and host memory storing computer code configured to perform an operation. The operation includes activating control queues between the host and the shared network adapter, issuing control commands to the shared network adapter using the control queues where the control commands configure a data plane for exchanging data between the host and the shared network adapter, activating data queues between the host and the shared network adapter after issuing the control commands, and upon identifying an error, suspending

the data queues while permitting the control queues to still transmit control plane data between the host and the shared network adapter.

[0006] According to one embodiment of the present invention a computer program product that includes a computer readable storage medium having computer-readable program code embodied therewith, the computer readable program code executable by one or more computer processors to perform an operation. The operation includes activating control queues between a host and a shared network adapter, issuing control commands to the shared network adapter using the control queues where the control commands configure a data plane for exchanging data between the host and the shared network adapter, activating data queues between the host and the shared network adapter after issuing the control commands, and upon identifying an error, suspending the data queues while permitting the control queues to still transmit control plane data between the host and the shared network adapter.

BRIEF DESCRIPTION OF THE DRAWINGS

- [0007]** Figure 1 illustrates a computing environment, according to one embodiment.
- [0008]** Figure 2 illustrates a shared resource environment, according to one embodiment.
- [0009]** Figure 3 illustrates data constructs for a data device, according to one embodiment.
- [0010]** Figure 4 is a flowchart for establishing data constructs for a data device, according to one embodiment.
- [0011]** Figure 5 illustrates a queue description record, according to one embodiment.
- [0012]** Figure 6 illustrates a queue descriptor array entry, according to one embodiment.
- [0013]** Figures 7A and 7B illustrate TX and RX storage block page entries, according to one embodiment.
- [0014]** Figure 8 illustrates a control program queue index, according to one embodiment.
- [0015]** Figure 9 illustrates a RX completion queue entry, according to one embodiment.
- [0016]** Figure 10 illustrates a shared resource environment, according to one embodiment.

[0017] Figure 11 is a flowchart for activating control queues and data queues in a data device, according to one embodiment.

[0018] Figure 12 is a state machine for activating control queues and data queues in a data device, according to one embodiment.

[0019] Figure 13 is a flowchart for configuring a data device, according to one embodiment.

[0020] Figure 14 illustrates a data device initialization flow, according to one embodiment.

[0021] Figure 15 is a flowchart for activating control queues and data queues in a data device, according to one embodiment.

[0022] Figure 16 is a flowchart of a method for transmitting a command using the control queues, according to one embodiment.

DETAILED DESCRIPTION

[0023] The embodiments herein describe enabling communication between a host and a shared network adapter. Data communication can be established first establishing control queues which are then used to establish data queues. For example, control queues are part of a control plane and data queues that are part of a data plane used to transmit data between the host and the shared network adapter. Advantageously, this makes it much easier to synchronize the data and control planes relative to using two separate data devices.

[0024] In one embodiment, when configuring the data device, the host first establishes the data and control queues but only activates the control queues. The control queues can then be used to transmit control commands from the host to the shared network adapter to configure the data plane. Once the data plane is configured, the host can then activate the data queues and the host and adapter can exchange data regarding TX and RX packets using the data queues.

[0025] In one embodiment, the host and the network adapter can deactivate the data planes, but still keep the control plane enabled. For example, the network adapter may experience an error and deactivate the data queues while the host and network adapter are still free to exchange control messages using the control queues. The network adapter can then restart the data plane once it has recovered. In this manner, the network adapter and the host can deactivate (or pause) the data plane and the data queues during operation without negatively impacting the control plane.

[0026] The descriptions of the various embodiments of the present invention have been presented for purposes of illustration, but are not intended to be exhaustive or limited to the embodiments disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope of the described embodiments. The terminology used herein was chosen to best explain the principles of the embodiments, the practical application or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments disclosed herein.

[0027] In the following, reference is made to embodiments presented in this disclosure. However, the scope of the present disclosure is not limited to specific described embodiments. Instead, any combination of the following features and elements, whether related to different embodiments or not, is contemplated to implement and practice contemplated embodiments. Furthermore, although embodiments disclosed herein may achieve advantages over other possible solutions or over the prior art, whether or not a particular advantage is achieved by a given embodiment is not limiting of the scope of the present disclosure. Thus, the following aspects, features, embodiments and advantages are merely illustrative and are not considered elements or limitations of the appended claims except where explicitly recited in a claim(s). Likewise, reference to "the invention" shall not be construed as a generalization of any inventive subject matter disclosed herein and shall not be considered to be an element or limitation of the appended claims except where explicitly recited in a claim(s).

[0028] Various aspects of the present disclosure are described by narrative text, flowcharts, block diagrams of computer systems and/or block diagrams of the machine logic included in computer program product (CPP) embodiments. With respect to any flowcharts, depending upon the technology involved, the operations can be performed in a different order than what is shown in a given flowchart. For example, again depending upon the technology involved, two operations shown in successive flowchart blocks may be performed in reverse order, as a single integrated step, concurrently, or in a manner at least partially overlapping in time.

[0029] A computer program product embodiment ("CPP embodiment" or "CPP") is a term used in the present disclosure to describe any set of one, or more, storage media (also called "mediums") collectively included in a set of one, or more, storage devices that collectively include machine readable code corresponding to instructions and/or data for performing computer operations specified in a given CPP claim. A "storage device" is any tangible device that can retain and store instructions for use by a computer processor. Without limitation, the computer readable storage medium may be an electronic storage medium, a magnetic storage medium, an optical storage medium, an electromagnetic storage medium, a semiconductor storage medium, a mechanical storage medium, or any suitable combination of the foregoing. Some known types of storage devices that include these mediums include: diskette, hard disk, random access memory (RAM), read-only memory (ROM), erasable programmable

read-only memory (EPROM or Flash memory), static random access memory (SRAM), compact disc read-only memory (CD-ROM), digital versatile disk (DVD), memory stick, floppy disk, mechanically encoded device (such as punch cards or pits / lands formed in a major surface of a disc) or any suitable combination of the foregoing. A computer readable storage medium, as that term is used in the present disclosure, is not to be construed as storage in the form of transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide, light pulses passing through a fiber optic cable, electrical signals communicated through a wire, and/or other transmission media. As will be understood by those of skill in the art, data is typically moved at some occasional points in time during normal operations of a storage device, such as during access, de-fragmentation or garbage collection, but this does not render the storage device as transitory because the data is not transitory while it is stored.

[0030] Computing environment 100 contains an example of an environment for the execution of at least some of the computer code involved in performing the inventive methods, such as establish data constructs for a data device 195 which permits the operating system 122 to communicate with a shared adapter interface (which is discussed in more detail in the later figures). In addition to the data device 195, computing environment 100 includes, for example, computer 101, wide area network (WAN) 102, end user device (EUD) 103, remote server 104, public cloud 105, and private cloud 106. In this embodiment, computer 101 includes processor set 110 (including processing circuitry 120 and cache 121), communication fabric 111, volatile memory 112, persistent storage 113 (including operating system 122 and data device 195, as identified above), peripheral device set 114 (including user interface (UI) device set 123, storage 124, and Internet of Things (IoT) sensor set 125), and network module 115. Remote server 104 includes remote database 130. Public cloud 105 includes gateway 140, cloud orchestration module 141, host physical machine set 142, virtual machine set 143, and container set 144.

[0031] COMPUTER 101 may take the form of a desktop computer, laptop computer, tablet computer, smart phone, smart watch or other wearable computer, mainframe computer, quantum computer or any other form of computer or mobile device now known or to be developed in the future that is capable of running a program, accessing a network or querying a database, such as remote database 130. As is well understood in the art of computer technology, and depending upon the technology, performance of a computer-implemented method may be distributed among multiple computers and/or between multiple locations. On the other hand, in this presentation of computing environment 100, detailed discussion is focused on a single computer, specifically computer 101, to keep the presentation as simple as possible. Computer 101 may be located in a cloud, even though it is not shown in a cloud in Figure 1. On the other hand, computer 101 is not required to be in a cloud except to any extent as may be affirmatively indicated.

[0032] PROCESSOR SET 110 includes one, or more, computer processors of any type now known or to be developed in the future. Processing circuitry 120 may be distributed over multiple packages, for example, multiple, coordinated integrated circuit chips. Processing circuitry 120 may implement multiple processor threads and/or multiple processor cores. Cache 121 is memory that is located in the processor chip package(s) and is typically used for data or code that should be available for rapid access by the threads or cores running on processor set 110. Cache memories are typically organized into multiple levels depending upon relative proximity to the processing circuitry. Alternatively, some, or all, of the cache for the processor set may be located "off chip." In some computing environments, processor set 110 may be designed for working with qubits and performing quantum computing.

[0033] Computer readable program instructions are typically loaded onto computer 101 to cause a series of operational steps to be performed by processor set 110 of computer 101 and thereby effect a computer-implemented method, such that the instructions thus executed will instantiate the methods specified in flowcharts and/or narrative descriptions of computer-implemented methods included in this document (collectively referred to as "the inventive methods"). These computer readable program instructions are stored in various types of computer readable storage media, such as cache 121 and the other storage media discussed below. The program instructions, and associated data, are accessed by processor set 110 to control and direct performance of the inventive methods. In computing environment 100, at least some of the instructions for performing the inventive methods may be stored in persistent storage 113.

[0034] COMMUNICATION FABRIC 111 is the signal conduction path that allows the various components of computer 101 to communicate with each other. Typically, this fabric is made of switches and electrically conductive paths, such as the switches and electrically conductive paths that make up busses, bridges, physical input / output ports and the like. Other types of signal communication paths may be used, such as fiber optic communication paths and/or wireless communication paths.

[0035] VOLATILE MEMORY 112 is any type of volatile memory now known or to be developed in the future. Examples include dynamic type random access memory (RAM) or static type RAM. Typically, volatile memory 112 is characterized by random access, but this is not required unless affirmatively indicated. In computer 101, the volatile memory 112 is located in a single package and is internal to computer 101, but, alternatively or additionally, the volatile memory may be distributed over multiple packages and/or located externally with respect to computer 101.

[0036] PERSISTENT STORAGE 113 is any form of non-volatile storage for computers that is now known or to be developed in the future. The non-volatility of this storage means that the stored data is maintained regardless of whether power is being supplied to computer 101 and/or directly to persistent storage 113. Persistent storage 113 may be a read only memory (ROM), but typically at least a portion of the persistent storage allows writing of data, deletion of data and re-writing of data. Some familiar forms of persistent storage include magnetic disks and solid state storage devices. Operating system 122 may take several forms, such as various known proprietary operating systems or open source Portable Operating System Interface-type operating systems that employ a kernel. The code included in the data device 195 typically includes at least some of the computer code involved in performing the inventive methods.

[0037] PERIPHERAL DEVICE SET 114 includes the set of peripheral devices of computer 101. Data communication connections between the peripheral devices and the other components of computer 101 may be implemented in various ways, such as Bluetooth connections, Near-Field Communication (NFC) connections, connections made by cables (such as universal serial bus (USB) type cables), insertion-type connections (for example, secure digital (SD) card), connections made through local area communication networks and even connections made through wide area networks such as the internet. In various embodiments, UI device set 123 may include components such as a display screen, speaker, microphone, wearable devices (such as goggles and smart watches), keyboard, mouse, printer, touchpad, game controllers, and haptic devices. Storage 124 is external storage, such as an external hard drive, or insertable storage, such as an SD card. Storage 124 may be persistent and/or volatile. In some embodiments, storage 124 may take the form of a quantum computing storage device for storing data in the form of qubits. In embodiments where computer 101 is required to have a large amount of storage (for example, where computer 101 locally stores and manages a large database) then this storage may be provided by peripheral storage devices designed for storing very large amounts of data, such as a storage area network (SAN) that is shared by multiple, geographically distributed computers. IoT sensor set 125 is made up of sensors that can be used in Internet of Things applications. For example, one sensor may be a thermometer and another sensor may be a motion detector.

[0038] NETWORK MODULE 115 is the collection of computer software, hardware, and firmware that allows computer 101 to communicate with other computers through WAN 102. Network module 115 may include hardware, such as modems or Wi-Fi signal transceivers, software for packetizing and/or de-packetizing data for communication network transmission, and/or web browser software for communicating data over the internet. In some embodiments, network control functions and network forwarding functions of network module 115 are performed on the same physical hardware device. In other embodiments (for example, embodiments that utilize

software-defined networking (SDN)), the control functions and the forwarding functions of network module 115 are performed on physically separate devices, such that the control functions manage several different network hardware devices. Computer readable program instructions for performing the inventive methods can typically be downloaded to computer 101 from an external computer or external storage device through a network adapter card or network interface included in network module 115.

[0039] WAN 102 is any wide area network (for example, the internet) capable of communicating computer data over non-local distances by any technology for communicating computer data, now known or to be developed in the future. In some embodiments, the WAN 102 may be replaced and/or supplemented by local area networks (LANs) designed to communicate data between devices located in a local area, such as a Wi-Fi network. The WAN and/or LANs typically include computer hardware such as copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and edge servers.

[0040] END USER DEVICE (EUD) 103 is any computer system that is used and controlled by an end user (for example, a customer of an enterprise that operates computer 101), and may take any of the forms discussed above in connection with computer 101. EUD 103 typically receives helpful and useful data from the operations of computer 101. For example, in a hypothetical case where computer 101 is designed to provide a recommendation to an end user, this recommendation would typically be communicated from network module 115 of computer 101 through WAN 102 to EUD 103. In this way, EUD 103 can display, or otherwise present, the recommendation to an end user. In some embodiments, EUD 103 may be a client device, such as thin client, heavy client, mainframe computer, desktop computer and so on.

[0041] REMOTE SERVER 104 is any computer system that serves at least some data and/or functionality to computer 101. Remote server 104 may be controlled and used by the same entity that operates computer 101. Remote server 104 represents the machine(s) that collect and store helpful and useful data for use by other computers, such as computer 101. For example, in a hypothetical case where computer 101 is designed and programmed to provide a recommendation based on historical data, then this historical data may be provided to computer 101 from remote database 130 of remote server 104.

[0042] PUBLIC CLOUD 105 is any computer system available for use by multiple entities that provides on-demand availability of computer system resources and/or other computer capabilities, especially data storage (cloud storage) and computing power, without direct active management by the user. Cloud computing typically leverages sharing of resources to achieve coherence and economies of scale. The direct and active management of the computing resources of public cloud 105 is performed by the computer hardware and/or software of cloud

orchestration module 141. The computing resources provided by public cloud 105 are typically implemented by virtual computing environments that run on various computers making up the computers of host physical machine set 142, which is the universe of physical computers in and/or available to public cloud 105. The virtual computing environments (VCEs) typically take the form of virtual machines from virtual machine set 143 and/or containers from container set 144. It is understood that these VCEs may be stored as images and may be transferred among and between the various physical machine hosts, either as images or after instantiation of the VCE. Cloud orchestration module 141 manages the transfer and storage of images, deploys new instantiations of VCEs and manages active instantiations of VCE deployments. Gateway 140 is the collection of computer software, hardware, and firmware that allows public cloud 105 to communicate through WAN 102.

[0043] Some further explanation of virtualized computing environments (VCEs) will now be provided. VCEs can be stored as “images.” A new active instance of the VCE can be instantiated from the image. Two familiar types of VCEs are virtual machines and containers. A container is a VCE that uses operating-system-level virtualization. This refers to an operating system feature in which the kernel allows the existence of multiple isolated user-space instances, called containers. These isolated user-space instances typically behave as real computers from the point of view of programs running in them. A computer program running on an ordinary operating system can utilize all resources of that computer, such as connected devices, files and folders, network shares, CPU power, and quantifiable hardware capabilities. However, programs running inside a container can only use the contents of the container and devices assigned to the container, a feature which is known as containerization.

[0044] PRIVATE CLOUD 106 is similar to public cloud 105, except that the computing resources are only available for use by a single enterprise. While private cloud 106 is depicted as being in communication with WAN 102, in other embodiments a private cloud may be disconnected from the internet entirely and only accessible through a local/private network. A hybrid cloud is a composition of multiple clouds of different types (for example, private, community or public cloud types), often respectively implemented by different vendors. Each of the multiple clouds remains a separate and discrete entity, but the larger hybrid cloud architecture is bound together by standardized or proprietary technology that enables orchestration, management, and/or data/application portability between the multiple constituent clouds. In this embodiment, public cloud 105 and private cloud 106 are both part of a larger hybrid cloud.

[0045] One embodiment of a shared resource environment to incorporate and use one or more aspects of the present invention is described with reference to Figure 2. In one example, a shared resource environment 200 is based, for instance, on the z/Architecture® offered by International Business Machines Corporation, Armonk, N.Y., and includes, for instance, a System z® server, also offered by International Business Machines Corporation.

z/Architecture®, System z® and IBM® are registered trademarks of International Business Machines Corporation, Armonk, N.Y., U.S.A. Other names used herein may be registered trademarks, trademarks or product names of International Business Machines Corporation or other companies. While the z/Architecture® and System z® are used as illustrative examples, the embodiments herein are not limited to this architecture and can be applied to any suitable hardware system and operating system as described in Figure 1. For example, the embodiments herein can be applied to any computing system where multiple entities (e.g., LPARs or virtual machines (VM)) share a common adapter 210.

[0046] In this example, shared resource environment 200 includes a central processor complex (CPC) 202, having, for instance, one or more partitions or zones 204 (e.g., logical partitions LPAR L1-LPAR L3, which can also be referred to as VMs). Each logical partition has a resident operating system 206, which may differ for one or more of the logical partitions. That is, the operating systems 206 may be different types of operating systems. Although in this example, three logical partitions are described, other embodiments can include more or fewer logical partitions. Further, one or more of the partitions may not be executing an operating system, and/or may execute operating systems other than those described herein. Moreover, one LPAR or VM can have multiple operating systems.

[0047] One or more of the logical partitions are managed by a hypervisor 250. The hypervisor 250 enables the hardware to virtualize the LPARs.

[0048] Each LPAR is coupled to a shared network adapter 210. The adapter 210 includes a network interface card/controller (NIC) 212, which enables communication via an external network 214. External network 214 is coupled to the NIC 212 via a port 216. The network 214 may be used to communicate between the LPARs of the shared resource environment 200 or with processors of other processing environments over a local area network (LAN) or wide area network (WAN).

[0049] The adapter 210 can include software code (e.g., microcode) that communicates with the operating systems in the LPARs. Put differently, this code provides an interface to communicate between the LPARs and the rest of the components in the adapter 210 (e.g., the NIC 212). As discussed in more detail below, the LPARs also have data devices 195A-195C which serve as interfaces between a respective operating system in the LPARs and the shared network adapter 210. While Figure 2 illustrates each LPAR having one data device 195, a LPAR (or an operating system in the LPAR) can have multiple data devices 195 (e.g., one data device 195 dedicated to IPv4 traffic and another data device 195 dedicated to IPv6 traffic).

[0050] The adapter 210 also includes a plurality of data connections 218, each of which is coupled to one of the devices 195 within an LPAR. For instance, a data connection 218A is coupled to a data device 195A in LPAR 1; data connections 218B, 218C are coupled to data devices 195B, 195C, respectively, in LPAR 2; and a data connection 218D is coupled to a device 195D in LPAR 3. In one example, the data connections 218 are enhanced queued direct I/O (EQDIO) data connections. Further, the data connections 218 can be used to transmit both data for user applications as well as control data.

[0051] The device 195A is further coupled to an entity 222A (such as TCP/IP, the protocol stack for OS) in LPAR 1; data devices 195B, 195C are further coupled to entities 222B, 222C (such as Guest C1, Guest C2), respectively, in LPAR 2; and device 195D is further coupled to a virtual switch 224 in LPAR 3.

[0052] The virtual switch 224 enables further sharing of data among entities 226A, 226B and 226C (e.g., Guest E1, E2 and E3) of LPAR 3. The virtual switch 224 includes a plurality of ports 228A, 228B and 228C, each of which is coupled to a respective guest via a NIC 230A, 230B, and 230C, respectively. The virtual switch allows the guests coupled thereto to communicate with one another without using the adapter or the external network.

[0053] Figure 3 illustrates data constructs for a data device, according to one embodiment. The system 300 illustrates data constructs in host memory 305, hardware system area (HSA) 310, and the shared adapter 210. The host memory 305 can be the memory in the LPARs or VMs, while the shared adapter 210 can include code (e.g., microcode) and memory that serves as an interface between the NIC 212 and the LPARs (e.g., the hosts).

[0054] The data constructs in the host memory 305 include the data device 195 (which is also established in the shared adapter 210 in a process described in later figures), TX storage block page entries (SBPE) 320, a control program (CP) queue index 325, RX SBPEs 335, an RX completion queue 340, and an interrupt register 345. As discussed above, the data device 195 establishes the connection between the host and the shared adapter 210. Each LPAR (or each operating system in the LPAR) that wishes to use the NIC 212 may create its own data device 195. The data device 195 can include any number of data queues that rely on the data constructs illustrated in Figure 3 to facilitate communication between the host and the shared adapter 210. For example, there can be a configurable number of Data Queues which can be any mix of TX or RX Queues. In one embodiment, there is at least one Data Plane TX and one Data Plane RX queue. In addition, the data device 195 includes at least one pair of control queues (i.e., one Control Plane TX queue and one Control Plane RX queue).

[0055] The format of the TX SBPEs 320 and the RX SBPEs 335 will be discussed in more detail in Figures 7A and 7B, but in general, these entries contain pointers to packets that are being transmitted from the host (in the

case of TX SBPEs 320) to the network via the NIC 212 or pointers to packets that are received at the NIC 212 from the network (in the case of RX SBPEs 335). The TX SBPEs 320 can also contain pointers to control information which is used to pass configuration or debug types of information between the Host and the shared adapter 210. Each SBPE can indicate that the packet is stored in one location, or that the packet is stored in multiple different locations (e.g., the headers may be stored in one location, while the payload is stored in another location). Thus, the pointers in the SBPEs 320 and 335 can support either type of strategy for storing the TX and RX packets in memory. Note that in one embodiment, the SBPEs themselves can store the TX and RX packets, rather than having pointers to those packets.

[0056] The CP queue index 325 can indicate the number of TX SBPEs 320 that are ready to be processed by the shared adapter 210. For example, assume that the host has four packets it wants the NIC 212 to send on the network 214. The host can create four TX SBPEs 320 (e.g., TXSBPE 0-3) for those four packets and then update the CP queue index 325 to inform the shared adapter 210 that the host has four packets ready to be transmitted. The shared adapter 210 can then read the CP queue index 325, identify the four TX SBPEs 320, and fetch the packets to be transmitted by the NIC 212 using the information in the four TX SBPEs 320. More details of the CP queue index 325 are provided in Figure 8.

[0057] In one embodiment, each TX queue and RX queue include its own CP queue index 325 and adapter (ADP) ADP queue index 330.

[0058] The RX completion queue 340 map locations of Ethernet packets in an RX Data Buffer. In one embodiment, the shared adapter 210 uses the completion queue 340 to indicate new RX packet arrivals at the NIC 212. The details of the RX completion queue 340 are provided in Figure 9.

[0059] The interrupt register 345 is set by the shared adapter 210 when an interrupt is generated. In one embodiment, the register 345 includes an interrupt state (e.g., one byte) which the host can set to indicate when an interrupt is required. For example, if the host has finished processing all the received packets, the CP in the host can set the interrupt state to indicate it is idle. Thus, if the shared adapter 210 receives additional packets, it can use the interrupt state to determine whether it should then send an interrupt to the host. For example, if the interrupt state indicates that the host is not currently processing data in the RX queues, then the shared adapter 210 can send an interrupt to the host. However, if the interrupt state indicates the host is currently processing RX packets, an interrupt is not needed.

[0060] Moreover, in one embodiment, the register 345 can include a bit mask where each bit corresponds to a queue ID of a queue in the data device. The shared adapter 210 can use the bit mask to indicate which queues has data ready to be processed by the host. While there may be multiple queues for the data device 195, there may be only one interrupt register 345 for each data device 195.

[0061] The HSA 310 is a special memory area used for Hardware Configuration tables. In this example, the HSA 310 includes the ADP queue index 330. In one embodiment, the ADP queue index 330 is read/write by the adapter 210, and read only by the CP. The shared adapter 210 updates the ADP Queue Index 330. The TX ADP Queue Index is used by the adapter to indicate the completion of a control plane request or the completion of a packet transmission. Although Figure 3 illustrates one ADP queue index 330, in one embodiment, the HSA 310 may store a TX ADP queue index and an RX ADP queue index. During transmission, the shared adapter 210 gets TX completions and updates the TX ADP queue index. As such, the TX ADP queue index 330 can store the next TX SBPE Index to be consumed by the shared adapter 210. Moreover, updates by the shared adapter 210 to the ADP queue index 330 can be used to indicate the completion of TX Control or Data requests. The CP Queue Index 325 indicates the next entry which will contain a new TX Control or Data request.

[0062] For an RX ADP queue index, the shared adapter 210 updates this index to describe received packets. The index can indicate the next RX data SBPE 335 to be consumed by the adapter 210 and the next RX completion queue entry 340 to be produced by the adapter 210. The Data SBPE Index is updated when the adapter 210 has used all allocated space in a Data SBPE and ownership of the buffer space is returned to the CP.

[0063] In addition to the data device 195 and the NIC 212, the shared adapter 210 also includes an adapter initiative index 350. This data construct can be used by the host to interrupt or wake up the shared adapter 210 when the host has packets ready to be transmitted. The shared adapter 210 can use the CP queue index 325 to retrieve the TX SBPEs 320 and programs the NIC to fetch the correspond TX packets from host memory 305 using the pointers in the TX SBPEs 320. When the adapter initiative index 350 is used, the CP copies the contents of the CP queue index 325 into the adapter initiative index 350. The adapter 210 can then directly fetch the SBPEs corresponding to the Queue Index without fetching the CP queue index 325.

[0064] Figure 4 is a flowchart of a method 400 for establishing data constructs for a data device, according to one embodiment. For example, the method 400 can configure the shared adapter to use the various data constructs illustrated in Figure 3.

[0065] At block 405, the host establishes data constructs for the data device in host memory. These data constructs can include, for example, the TX SBPEs 320, the CP queue index 325, the RX SBPEs 335, the RX completion queue entries 340, and the interrupt register 345.

[0066] At block 410, the host transmits a queue description record (QDR) to the shared adapter 210. One example of a QDR is illustrated in Figure 5. For example, the host may use a different I/O protocol to communicate with the shared adapter before the control plane and data plane have been established. One example I/O protocol can be channel command word (CCW), which defines I/O operations used for communications with a channel subsystem. The CCW contains a channel command, such as read, write, or control, along with the data address of the data area involved. However, the embodiments herein are not limited to any particular I/O protocol so long as that protocol can transmit the QDR to the shared adapter.

[0067] At block 415, the shared adapter establishes TX and RX queues for the data service in the shared adapter using the QDR. In one embodiment, the shared adapter first establishes TX and RX control queues for establishing the control plane, and then establish TX and RX data queues for the data plane, after the control plane has been establish.

[0068] The information used to establish the control and data queues is described using Figure 5 as an example. The QDR 500 in Figure 5 includes a QFMT 505, a version 510, a queue descriptor size 515, TX queue count 520, RX queue count 525, a system queue control area (SQCA) 530, an interrupt reduction control (IRC) 535 (also referred to as simply "interrupt control"), queue information block (QIB) 540, and queue descriptor array entries 545.

[0069] The Queue Format (QFMT) 505 enables the use of multiple protocols, each using a specific format for their SBPEs, CP and ADP Queue Index areas, Interrupt Register, etc.

[0070] The version 510 enables the support for multiple versions of a specific QFMT. For example, Version 2 might support an expanded format of the SBPEs which is not supported in Version 1.

[0071] The queue descriptor size 515 tells the shared adapter 210 the size of the queue descriptor array entries 545.

[0072] The TX queue count 520 tells the shared adapter 210 how many TX queues the data device has. In one embodiment, the QDR contains a Queue Descriptor Entry for each of the Control Queues and each of the Data

Queues. For example, the RX Control Queue can be hardcoded as Queue ID 0 and the TX Control Queue as Queue ID 1.

[0073] As for Data Queues, the data device may have multiple TX queues where each queue can have a different priority. For example, for each pass, the shared adapter may only process at most 5 packets for one TX queue (even if it has more packets than that) but may process up to 10 packets for another, higher priority, TX queue in the data device.

[0074] The RX queue count 525 tells the shared adapter 210 how many RX queues the data device has, which can include a RX control queue and any number of RX data queues. Like the TX queues, the data device may include multiple RX data queues which may be assigned different priorities. The host may process the received packets differently depending on which RX queue they are in. One other use for multiple RX queues is the host can configure a specific RX queue for a special traffic type (e.g., ARP) therefore enabling the host to attach a special program specifically designed for that traffic type.

[0075] The SQCA 530 contains the address for the CP queue index 325 in host memory 305 in Figure 3. That is, the SQCA 530 contains the memory address of the CP queue index 325 so that the shared adapter 210 can read the index 325 to determine which of the TX SBPEs 320 in host memory 305 correspond to new TX packets. In one embodiment, each TX queue in the data device has its own set of TX SBPEs 320. As such, each TX queue can also have its own CP queue index 325. The SQCA 530 can contain the address of each CP queue index for the data device. In one embodiment, the SQCA 530 is a 256 byte area and each queue has a defined SQCA. For example, the first 8 bytes of the SQCA 530 is the Queue Index area associated with the defined queue. The SQCA Address in the QDR can be a pointer to a 4K Memory Area which contains 16 SQCAs. The QDR contains two of these pointers to represent a total of 32 possible queues (16 in each 4K page.)

[0076] In some embodiments, the QDR 500 may include multiple SQCAs 530 if a predefined data size (e.g., 64 bits) is not sufficient to store the addresses for each CP queue index (e.g., if the data drive has a bunch of TX queues).

[0077] The IRC 535 stores the memory address of the interrupt register 345 in the host memory 305 in Figure 3. That way, the shared adapter 210 can query this register 345 so it can interrupt the host when there are new received packets ready for it to process as discussed above.

[0078] The QIB 540 informs the shared adapter 210 of the properties of the queues in the data device. The QIB 540 can contain information which is global to all queues. This may include things like the Host and Adapter

Capabilities. The Host capabilities contain, e.g., the Cache Line size. The Adapter Capabilities can be protocol offloads, LAN speed, link aggregation functionality, etc.

[0079] In this example, the queue descriptor array entries 545 can include different amounts of data (e.g., have a varying size) which is why the QDR 500 includes the queue descriptor size 515. The details of the queue descriptor array entries 545 are described in Figure 6.

[0080] Figure 6 illustrates a queue descriptor array entry 545 in a QDR (e.g., the QDR 500 in Figure 5). In this example, the queue descriptor array entry 545 includes a queue ID 605 to identify the specific queue in the data device that corresponds to the entry 545 and a queue type 610 indicating whether the corresponding queue is a control queue, a RX queue, or TX queue, etc.

[0081] The queue descriptor array entry 545 also includes a Storage Block Table (SBT) 615, a Queue Format Record Information Block (QFRIB) 620, and a RX Ethernet Packet Completion Queue (EPCQ) 625.

[0082] The SBT 615 stores an address that maps the TX SBPEs or RX data SBPEs corresponding to the queue to locations in host memory. That way, the shared adapter 210 knows where in host memory the TX SBPEs 320 or RX data SBPEs 335 are located.

[0083] The QFRIB 620 stores an address that contain specific configuration information for the TX or RX queue type.

[0084] The RX EPCQ 625 stores an address that maps to the location of the RX completion queue entries 340. That way, the shared adapter 210 knows where in host memory the RX complete queue entries 340 are located.

[0085] In this manner, the QDR 500 in Figure 5 (which can include the queue descriptor array entry 545 in Figure 6) can be transmitted by the host to the shared adapter 210 so the shared adapter 210 can use the data constructs established by the host for the data device. That is, the shared adapter 210 can use the information in the QDR 500 to establish the TX and RX queues that permit the LPARs and the NIC to communicate.

[0086] Figure 7A illustrates a TX SBPE 320, according to one embodiment. That is, Figure 7A is one example of a TX SBPE 320 discussed above. As shown, the SBPE 320 includes an R bit 705, an I bit 710, a type 715, flags 720, extended flags 725, error code 730, length 735, and CP buffer address 740.

[0087] The I bit 710 identifies whether the packet represented by the SBPE 320 is immediate data where the SBPE stores the TX packet rather than storing a pointer to TX packet. This may be valid only for the data plane, and not for the control plane.

[0088] The type 715 identifies data type of the corresponding packet, such as 0x01 control packet or a 0x02 Ethernet packet, etc.

[0089] The flags 720 can be a chaining flag to "chain" together multiple SBPEs where the data packet is stored in different memory locations or indicate the packet is stored in contiguous memory locations. The flags 720 may indicate that an interrupt should be generated by the shared adapter when processing the corresponding packet. The flags 720 may also include an error flag, where the error code 730 can contain the specific error code.

[0090] The extended flags 725 can be reserved for use by the CP executing in the host.

[0091] The length 735 indicates the length of the Control Information or Data pointed to by the Buffer Address in the TX SBPE.

[0092] In one embodiment, the CP buffer address 740 supports any byte aligned address. However, the buffer address 740 may be restricted where the buffer address 740 plus the length 735 cannot cross a 4K boundary, and instead should use chaining in that instance.

[0093] Figure 7B illustrates a RX SBPE, according to one embodiment. That is, Figure 7B is one example of a RX SBPE 335 as introduced in Figure 3. As shown, the SBPE 335 includes a CP buffer address 750, a CP buffer address extender 760, and a reservation for CP flags 770.

[0094] The CP buffer address 750 and the (optional) CP buffer address extender 760 can store a 64 bit CP buffer address. For example, bits 0:31 of the address may be stored in the CP buffer address 750 while the remaining 32:63 bits of the address are stored in the CP buffer address extender 760.

[0095] In one embodiment, the low order bits (e.g., 12 of the lowest order bits) are reserved for CP flags. Reserving the low order bits forces the address in the SBPE to be 4K aligned. This is used to align the addresses with the Memory Page allocation algorithm which allocates memory in 4K blocks.

[0096] Figure 8 illustrates a CP queue index 325, according to one embodiment. The CP queue index 325 can indicate the number of TX SBPEs 320 that are ready to be processed by the shared adapter 210. The index 325 includes an initiative state 805, a SBPE index 810, and a completed SBPE index 815.

[0097] The initiative state 805 can tell the host whether the shared adapter is busy processing the TX SBPEs 320. That is, when a shared adapter is currently processing the TX SBPEs, it can update the initiative state 805 so the host (e.g., LPAR) knows that the shared adapter is currently processing the TX SBPEs. Thus, if the host has more packets ready to transmit (e.g., creates additional TX SBPEs 32), the host can query the initiative state 805, determine the shared adapter is currently processing other TX packets, and know it does not need to alert the adapter. When done processing the TX packets (i.e., the when the shared adapter has completed all outstanding work), the shared adapter can update the initiative state 805 to indicate it is done. Later, if the host has more TX packets to process, it can query the initiative state 805. If the state 805 indicates the shared adapter 210 is idle, the host updates the initiative state 805 as it posts the new work and interrupts the shared adapter 210 so the adapter 210 knows that more TX packets are ready to be sent.

[0098] The SBPE index 810 can be updated to initiate data transfer from the host to the shared adapter. The completed SBPE index 815 is updated to indicate the SBPEs that have been completed by the host and are ready to be processed by the shared adapter. For example, if the host filled in TX SBPEs 0-3, the host can update the completed SBPE index 815 to say "four" indicating that the host was filled in SBPEs 0-3 and SBPE 4 is the next SBPE the host will use. This tells the shared adapter 210 that SBPEs 0-3 are ready to be processed. Once the CP SBPE Index is updated, it transfers ownership of the TX SBPE(s) to the shared adapter. At this point, the TX SBPE(s) becomes read only to the CP.

[0099] In one embodiment, the host can also include a RX CP queue index. That is, while Figure 8 illustrates a TX CP queue index, a RX CP queue index can be used that includes an initiative state, a data SBPE index which is updated to allocate RX buffers, and a EPCQ SBPE index which is updated by the host to acknowledge RX packets. In one embodiment, the Initiative State in both the CP TX and RX Queue Index areas works exactly the same. The initiative states can tell the CP whenever the indexes are updated and if the shared network adapter needs initiative (e.g., if the shared adapter should be interrupted). For the CP RX Queue Index, the shared network adapter can periodically check for new Data SBPEs and EPCQs. In one example, the shared network adapter only sets the Initiative State to indicate initiative is required when one of these resources becomes unavailable. This prevents the shared network adapter from having to poll these areas.

[00100] Figure 9 illustrates a RX completion queue entry, according to one embodiment. The RX completion queue entry 340 maps locations of Ethernet packet in an RX Data Buffer. The entry 340 includes a T bit 905, a starting alignment index 910, an RX data SBPE index 915, flags 920, and a length 925.

[00101] The T bit 905 is a toggle bit which will be toggled by the shared network adapter each time the EPCQ wraps. This bit can be used by the CP to detect new work without having to access the ADP RX Queue Index area.

[00102] The starting Alignment Index 910 identifies a packet's starting location within a RX Buffer in specified alignment units (e.g., 256 bytes). In one embodiment, the alignment index alignment unit is based on the CP Cache line size. This can be a programmable value controlled by the QIB.

[00103] The RX Data SBPE Index 915 contains the SBPE Index associated with the RX Storage Block Page Entries 335 where the Ethernet packet was placed.

[00104] The flags 920 can contain error and Ethernet packet type flags.

[00105] The length 925 can specify the actual Ethernet frame length from a NIC RX Descriptor. This enables the CP to know the actual size of the Ethernet packet provided by the NIC.

[00106] In one embodiment, each EPCQ entry represents a specific Ethernet packet arrival.

[00107] Figure 10 illustrates a shared resource environment, according to one embodiment. The environment 1000 includes many of the same components and elements discussed in previous figures as indicated by using the same reference numbers. A detailed description of those components will not be repeated here.

[00108] The environment 1000 illustrates a control plane 1005 and a data plane 1015 for the data device 195. The control plane 1005 includes control queues 1010 which the host and the shared adapter 210 can use to exchange control data (e.g., control commands) while the data plane 1015 include data queues 1020 which the host and shared adapter 210 use to exchange data regarding TX and RX network packets.

[00109] In one embodiment, the control queues 1010 and the data queues 1020 are examples of the data connections 218 in Figure 2 which can be, for example, QDIO data connections. As discussed in more detail below, before configuring the QDIO connections, the data device 195 may first use a different protocol (e.g., a channel command word (CCW) protocol) to facilitate communication between the host and the shared adapter 210. The data device 195 can then be configured to use a different protocol (e.g., enhanced QDIO) which supports the control queues 1010 and the data queues 1020.

[00110] Advantageously, the data device 195 can include both control queues 1010 and data queues 1020 unlike prior solutions where there is one data device that establishes the control plane (with control plane queues) and another data device that establishes the data plane (with data plane queues). Synchronizing the two data

devices was difficult. However, this is avoided in Figure 10 since the data device 195 includes both the control plane 1005 and the data plane 1015 which make synchronization easier. The interaction between the control plane 1005 and the data plane 1015 is discussed in more detail in Figures 11 and 12 below.

[00111] Moreover, an “activate data queues” control command enables the architecture for data queues. This enables communication to be established between the OS and the shared Network adapter for the data queues. In one embodiment, the system has the capability to activate a single data queue.

[00112] The environment 1000 also includes a CP 1025 in the host memory 305 which includes microcode 1030. The microcode 1030 can serve as an intermediary layer situated between the processor complex and the programmer-visible instruction set architecture (ISA) of the host. In one embodiment, the microcode 1030 provides a common interface for the operating systems in the host to the shared adapter 210. That way, different operating systems (e.g., different types of operating systems) executing in the shared environment 1000 can use the same instructions provided by the microcode 1030 to communicate with the shared adapter 210.

[00113] Figure 11 is a flowchart of a method 1100 for activating control queues and data queues in a data device, according to one embodiment. At block 1105 the host (or the microcode 1030 in Figure 10) activates the control queues between the host and the shared network adapter. In one embodiment, the host may have already established (or configured) both the control queues and the data queues of the data device which enable an operating system in the host to communicate with the shared adapter; however the host may only activate the control queues. That is, at block 1105, the data queues may be deactivated and not permitted to transmit data.

[00114] The host may perform several operations in order to establish the control queues and the data queues for the data device. The operations that may be performed before block 1105 in order to establish the queues will be discussed in Figure 13 below.

[00115] At block 1110, the host (or the microcode) issues control commands to the shared network adapter using the control queues. For example, at block 1110 the host can issue the protocol control commands used to configure the communications protocol to be used by the data plane. For instance, the host may issue IP assist primitives that are used for Control types of information. The IP assist primitives can be used by the Host to configure the various Data Queues so when the Activate Data Queues is issued, the underlying hardware and/or software is properly configured to Transmit packets from the proper TX Data Queue(s) and to receive packets into the proper RX Data Queue(s).

[00116] In one embodiment, the control plane is asynchronous where the host (or the CP) has the option to either execute a single control command and wait for its reply before issuing another command, or issue multiple commands (SBPEs) before processing any replies returned by the adapter. The method used will depend on the type of command issued, whether the multiple commands are completely independent of each other, or the CP's preference.

[00117] To keep track of a specific control operation, the CP can assign a unique sequential sequence number in each command request issued on the control plane's output queue. The adapter in turn returns this same sequence number back in the reply with the results of the request via the control plane's input queue. This allows the CP to correlate a reply received on its control plane's input queue to a specific request previously issued via its control plane's output queue.

[00118] Some example control commands and IP assist primitives that can be used to configure the data plane include:

- Storing subchannel data where the CP retrieves any subchannel QDIO characteristics, which may have changed or altered by the hypervisor after successfully activating the control plane. In one embodiment, the CP determines whether the current environment requires either a "Subchannel ID" (SID) or "Subchannel Token".
- An exchange adapter control command request which initiates communications to enable the control plane by informing the adapter of the CP managing the interface and its characteristics.
- A set group parameters request which can be used when grouping multiple adapters into a logical group. This command is typically used when bonding multiple adapters to a single logical configuration.
- A start LAN request. If the NIC port is operational when receiving this command, the adapter responds with a start LAN Reply at which point the CP can continue with its data plane setup.
- A set adapter parameters request which gets the supported hardware adapter commands that can be issued on this interface using SETADAPTERPARMS IP Assist Protocol Command. The hardware commands give the CP the ability to change the operational parameters of the NIC card.
- Commands to register one or more MAC addresses for the interface connection.

- Commands to define a single virtual MAC address. This can include a reply that sets a primitive return code along with a VLAN return code for each VLAN ID set.
- Commands that define a single or multiple VLAN IDs with a single MAC address. These commands can include a reply that set a primitive return code along with a VLAN return code for each VLAN ID set.
- Commands that define multiple virtual Ethernet MAC addresses in a single primitive. In one embodiment, each Virtual MAC in the Primitive has only a single VLAN specified. A reply primitive can have a unique return code for each Virtual MAC address and VLAN ID specified in a request.
- Commands to delete a block of MAC address that can remove multiple Virtual Ethernet MAC addresses using, e.g., a single primitive. A reply can be used to set the Primitive Return Code along with the specific Return Code associated with each specified Virtual MAC Address to be deleted.
- Commands to define a Group MAC Address where a single Group MAC can be defined to multiple VLAN IDs in the case of a Hypervisor. Another command primitive can be used to delete a Group MAC and its associated Multicast IP Address if applicable.
- Commands to register to receive traffic for the specified VLAN ID. A reply primitive can set the Primitive return code along with a Return Code.
- Commands to set group parameters (SETGROUPPARMS) for a logical group of connections.

[00119] These are just a few examples of possible adapter control commands and IP assist commands that can be transmitted on the control plane using the control queues in order to configure the data plane of the data device.

[00120] At block 1115, the host (or CP and the microcode) activates the data queues between the host and the shared network adapter. That is, once the Protocol Control Commands to setup the data plane have completed, the CP asks the adapter to start data flowing in both directions on the data plane. For example, the CP can send (using the control queues) a command that instructs the adapter to activate the defined data plane queues for inbound and outbound data transmissions. In one embodiment, this command is a final indication from the CP to the adapter that it is ready to accept and transmit data on the data plane. In the case of an Ethernet IP protocol, this can include unicast, multicast, and broadcast packet transmissions.

[00121] At block 1120, the OS in the host and the shared adapter exchange data corresponding to TX and RX packets using the data queues. This can include the data in the TX and RX data packets, the data in the SBPEs, the data in the CP queue index, the data in the RX completion queue entries, the data in the ADP queue index, etc.

[00122] Figure 12 is a state machine 1200 for activating control queues and data queues in a data device, according to one embodiment. At state 1205, the host (or the CP in the host) activates the control plane. This was described in detail in block 1105 in Figure 11.

[00123] At state 1210, the host/CP resets the interface and only performs adapter management commands. That is, the data plane and data queues are not active. The control plane can enter the state 1210 whenever the host executes the exchange adapter control command request mentioned above which initiates communications to enable the control plane by informing the adapter of the CP managing the interface and its characteristics. In one embodiment, this command is the first control command issued by the CP to place the control plane in the state 1210. In one embodiment, no other control command is accepted by the shared adapter until it is in the state 1210.

[00124] In one embodiment, a CP executing the exchange adapter control command request while in any other state, forces the adapter to reset the data plane. This action places the control plane back in the state it was originally in when the control plane was first activated. The inbound and outbound data transmissions on the data plane are deactivated and any Adapter and Protocol Control settings are reset. Issuing the exchange adapter control command resets all the queues (Control and Data) and places the Control Plane in a state where the only control command that can be executed is the start LAN request/command.

[00125] In one embodiment, the control plane remains in the state 1210 until the start LAN request mentioned above completes successfully. While in the state 1210, in one embodiment, only Adapter Control Commands may be issued by the OS. In one embodiment, while in the state 1210, only the Start LAN primitive can be issued which in turn activates the Control Queues to be able to accept the Configuration control primitives over the Control Queues.

[00126] A CP or adapter transitions the control plane into the state 1220 when it is ready to configure the communication protocol to be used by the data plane. When a start LAN Adapter Control Command completes successfully, the OS and the adapter start the LAN. Once the OS transitions to the Start LAN State, it can start issuing the Protocol Control Commands to configure the communications protocol to be used by the data plane. In

the case of an Ethernet IP interface, the CP can issue the IP Assist Primitives used for data transfer and any functions being offloaded to the adapter. When the Start LAN Control Command completes, the shared network adapter signals the OS that the shared network adapter is ready to receive protocol control commands to configure the communications protocol to be used on the data plane. It basically opens the control queues for communication using the protocol control commands and works similar to the "Activate Data Queues" command for the data queues. Prior to the Start LAN completion, in one embodiment, only the IDX (Exchange Adapter) and Start LAN control commands are accepted on the Control Queues.

[00127] The adapter can start the LAN after it recovers from an error which requires the protocol settings configured previously by the CP to be reset. Typically, this is when a NIC port transitions from operational to nonoperational due to a loss-of-light issue on the physical port. In this case, the adapter previously changed the control plane to the state 1215 when the port went inoperable and the adapter issued a stop LAN control command to the OS. The shared network adapter can reset back to the Start LAN State anytime by issuing the StopLAN Primitive. This purges the configurations previously defined by control primitives issued after the last successful StartLAN control primitive.

[00128] When a start LAN command successfully completes, the CP may reissue the Protocol Control Commands needed to restart the data plane after the error. To resume transmissions on the data plane, the CP can issue another activate data queues control command to enter the state 1225 where the data plane is fully operational.

[00129] In addition to the data plane (and data queues) being operational in state 1225, while the control plane is in state 1225, the CP may issue any type of control command to modify its running data plane environment.

[00130] If for any reason an adapter or CP wants to pause data transfer on the data plane queues without affecting the current protocol configuration, either the adapter or CP issues a deactivate data queues Adapter Control Command. For example, the adapter may experience an error that it wants to recover from and needs to suspend the data queues until it does. On successful completion of the request, all data plane transmissions cease and the control plane transitions into the state 1230 state. The control plane remains in this state until it transitions back to be fully operational again with an adapter or CP issuing an activate data queues request, or back to a state requiring further reconfiguration like the states 1210 or 1215. While the control plane is in the state 1230, in one embodiment, the CP may only issue Adapter Control Commands to the adapter.

[00131] Further, issuing the exchange adapter control command request at any time allows a CP to reset itself and start over whenever an unexpected "Protocol Violation" return code is received from the adapter. This action allows the CP to reset the control plane into a known state so it can start over with data plane initialization again. Issuing the exchange adapter control command resets all the queues (Control and Data) and places the Control Plane in a state where the only control command that can be executed is the start LAN request/command.

[00132] Figure 13 is a flowchart of a method 1300 for configuring a data device, according to one embodiment. In one embodiment, the method 1300 is performed before block 1105 in Figure 11. For example, the method 1300 can be used to establish the data connections (e.g., the control and data queues) before they are activated as described in the method 1100.

[00133] In one embodiment, before starting the method 1300, the OS in the host may cleanup or terminate any previous interface connections.

[00134] At block 1305, the OS in the host determines the type of the data device. For example, the OS may determine (or sense) whether the data device is of a type that can support the data constructs and data connections discussed above in the previous figures. If the OS determines at block 1310 the data device is not compatible (e.g., the data device is an older version), then the method 1300 ends. For example, previous data devices did not support having both control queues and data queues as shown in Figure 10.

[00135] However, if the data device is compatible with the data constructs and the techniques discussed in the previous figures, the method 1300 proceeds to block 1315 where the OS determines the capabilities of the shared network adapter. For example, the OS can determine the maximum number of queues supported by the adapter and any other related characteristics.

[00136] At block 1320, the OS generates a QDR for the shared network adapter. For instance, by determining the characteristics of the adapter, the OS can generate a QDR for configuring the queues supported by the adapter as discussed above in Figure 5.

[00137] At block 1325, the OS establishes control and data queues in the shared network adapter. For example, as discussed in Figure 4, the adapter can use the data in the QDR to establish the queues and to identify the data constructs in the host memory (e.g., the TX/RX SBPEs, the CP queue index, the TX completion queue entries, etc.). Further, the OS may also allocate and set device controls to enable the interface connection for adapter interruptions.

[00138] The method 1300 can then proceed to block 1105 to begin the method 1100 in Figure 11.

[00139] Figure 14 illustrates a data device initialization flow 1400, according to one embodiment. The flow 1400 is divided into three parts: a CCW mode 1405, a control plane active state 1420, and a data plane active state 1425. The CCW mode 1405 is used to communicate with the data device between the control plane and data planes are inactive. CCW is just one example I/O protocol that could be used. Any I/O protocol that defines I/O operations that can be used for communicating with a channel subsystem can be used.

[00140] In the CCW mode 1405, at block 1410, the host uses a QDR (e.g., the QDR 500 discussed in Figure 5 above) to establish the control plane queues. In one embodiment, the shared adapter first sets up TX and RX control queues for establishing the control plane using the information in the QDR 500. Put differently, the information in the QDR 500 is used to configure the TX and RX control queues in the data device that then can be used by the host and the shared network adapter to exchange control commands and requests (e.g., IP assist primitives).

[00141] At block 1415, the host activates the TX and RX control queues that were established at block 1410. This results in the control plane being active, which moves the flow 1400 into the control plane active state 1420. There, a state machine (e.g., the state machine 1200 in Figure 12) can be used to establish, configure, and then activate data queues for a data plane. Put differently, the state machine 1200 describes using the control plane in the data device in order to activate the data plane of the data device that permits the OS in the host and the shared network adapter to exchange data.

[00142] In one embodiment, activating the data queues can include a Start LAN command. This may be the first IP assist primitive sent by the OS. In response, the shared network adapter starts the LAN. Once started, other control commands may be exchanged. These commands can be protocol management configuration commands, such as the one discussed at block 1110 of the method 1100 in Figure 11.

[00143] Once the data plane is active, the flow 1400 moves to the data plane active state 1425. That is, both the control plane and the data plane are active. However, at block 1430, the host or the shared network adapter deactivates the data queues (e.g., suspends the data plane) due to an error. Examples of errors where the OS or the shared network adapter deactivate the data queues will be discussed in more detail in Figure 15.

[00144] In one embodiment, deactivating the data queues deactivates the data plane but its prior configuration remains intact. That is, deactivating the data queues does not necessarily reset its configuration. For example, the data queues could be deactivated to give the time for the shared network adapter to recover from an internal

error. Once it recovers, the shared adapter can resume the data queues without having to reissue any of the adapter and protocol management commands used to configure the data plane (since it retained its previous configuration). However, for other errors, the OS or the shared network adapter may reset the configuration of the data plane so it should be reconfigured before the data queues are resumed. This is discussed in the next block of the flow 1400.

[00145] At block 1435, as part of deactivating the data queues, the LAN may be stopped. In one embodiment, stopping the LAN transitions to a pending state that erases (or resets) the prior or current configuration of the data plane. In this scenario, the host and the shared network adapter may reconfigure the data plane before the data queues are resumed.

[00146] Figure 15 is a flowchart for activating control queues and data queues in a data device, according to one embodiment. At block 1505 the host (or the CP 1025 in Figure 10) activates the control queues between the host and the shared network adapter. In one embodiment, the host may have already established (or configured) both the control queues and the data queues of the data device which enable an operating system in the host to communicate with the shared adapter; however the host may only activate the control queues. That is, at block 1505, the data queues may be deactivated and not permitted to transmit data.

[00147] The host may perform several operations in order to establish the control queues and the data queues for the data device. Example operations that may be performed before block 1505 in order to establish the queues were discussed above in Figure 13.

[00148] At block 1510, the host or the CP issues control commands to the shared network adapter using the control queues. For example, at block 1510 the host can issue the protocol control commands used to configure the communications protocol to be used by the data plane. For instance, the host may issue IP assist primitives that are used for control types of information. The IP assist primitives can be used by the Host to configure the various Data Queues so when the Activate Data Queues command is issued, the underlying hardware and/or software is properly configured to Transmit packets from the proper TX Data Queue(s) and to receive packets into the proper RX Data Queue(s). Examples of these IP assist primitives (as well as adapter control commands) were discussed at block 1110 of method 1100 in Figure 11. The control commands can also include the adapter and protocol management commands discussed in Figure 12.

[00149] In one embodiment, the control plane is asynchronous where the host (or the CP) has the option to either execute a single control command and wait for its reply before issuing another command, or issue multiple

commands (SBPEs) before processing any replies returned by the adapter. The method used will depend on the type of command issued, whether the multiple commands are completely independent of each other, or the CP's preference.

[00150] To keep track of a specific control operation, the CP can assign a unique sequential sequence number in each command request issued on the control plane's output queue. The adapter in turn returns this same sequence number back in the reply with the results of the request via the control plane's input queue. This allows the CP to correlate a reply received on its control plane's input queue to a specific request previously issued via its control plane's output queue.

[00151] At block 1515, the host or CP activates the data queues between the host and the shared network adapter. That is, once the Adapter and Protocol Management Commands to setup the data plane have completed, the CP asks the adapter to start data flowing in both directions on the data plane. For example, the CP can send (using the control queues) a command that instructs the adapter to activate the defined data plane queues for inbound and outbound data transmissions. In one embodiment, this command is a final indication from the CP to the shared network adapter that it is ready to receive and transmit data on the data plane. In the case of an Ethernet IP protocol, this can include unicast, multicast, and broadcast packet transmissions.

[00152] At block 1520 the OS, CP, or shared network adapter identify an error. The error can include a NIC port transitioning from operational to nonoperational due to a loss-of-light issue on the physical port, an error experienced by the adapter, a protocol violation code, and the like. In one embodiment, the error may be any error where the data plane should be suspended.

[00153] At block 1525, the CP or OS suspends the data queues while permitting the control queues to still transmit control plane data. In one embodiment, the CP pauses data transfer on the data plane queues without affecting the current protocol configuration by issuing a deactivate data queues Adapter Control Command. For example, all data plane transmissions cease and the control plane may transition into the state 1230 shown in Figure 12. The control plane remains in this state until it transitions back to be fully operational again with an adapter or CP issuing an Activate Data Queues request, or back to a state requiring further reconfiguration like the states 1210 or 1215 shown in Figure 12.

[00154] In addition to suspending the data queues, the CP can reset the control plane into a known state so it can start over with data plane initialization again. For example, issuing the exchange adapter control command resets all the queues (Control and Data) and places the Control Plane in a state where the only control command

that can be executed is the start LAN request/command. In one embodiment, the data plane is reconfigured before it can be resumed.

[00155] Figure 16 is a flowchart of a method 1600 for transmitting a command using the control queues, according to one embodiment. At block 1605, the OS in the host retrieves a local copy of the next available SBPE to be used for a control outbound operation. In one embodiment, the operation is a control command that the host wants to send to the shared adapter.

[00156] At block 1610, the CP or shared adapter returns the current CP Queue Index register from the control queue's SQCA. The SQCA was discussed in detail in Figure 5 above.

[00157] At block 1615, the OS generates the control request. For example, the OS can generate an Adapter or Protocol Management Command in the SBPE (Local CP Queue Index) on the control plane's output queue. Examples of these control commands were discussed above at block 1110 of the method 1100.

[00158] At block 1620, the OS increments the local CP Queue Index. This will setup the local index for the next available SBPE for an outbound transmission. In this example, the "outbound transmission" over the Control Queue is from the perspective of the O/S sending an Adapter or Protocol Management Command to the shared network adapter.

[00159] At block 1625, the OS stores the updated local CP Queue Index with a Set CP Queue Index Controls (SCPQC) instruction. The SCPQC Instruction is used to manage the ring's CP Queue Index for sending data out on its output queue. In one embodiment, storing the updated Local CP Queue Index with the SCPQC Instruction gives the control command in SBPE 0 to the shared adapter for transmission.

[00160] At block 1630, the CP or the shared adapter sets the new CP Queue Index Register in the queue's SQCA. Optionally, the CP also provides initiative to the adapter to process the control queue.

[00161] At block 1635, the shared adapter generates a reply to the control request generated by the OS. Also, the shared adapter places the reply on the next available SBPE on the control plane's input queue. The shared adapter can also indicate work is pending for that input queue in IRC (i.e., interrupt control). The CP uses the IRC to determine the input queues which have outstanding work to process. If the IRC is in a "No Interrupt State", then the method stops here. Otherwise, the interface is placed in the No Interrupt State in the IRC.

[00162] While the foregoing is directed to embodiments of the present invention, other and further embodiments of the invention may be devised without departing from the basic scope thereof, and the scope thereof is determined by the claims that follow.

CLAIMS

What is claimed is:

1. A method comprising:
activating control queues between a host and a shared network adapter;
issuing control commands to the shared network adapter using the control queues, wherein the control commands configure a data plane for exchanging data between the host and the shared network adapter;
activating data queues between the host and the shared network adapter after issuing the control commands; and
upon identifying an error, suspending the data queues while permitting the control queues to still transmit control plane data between the host and the shared network adapter.
2. The method of claim 1, wherein the control queues are part of a control plane in a data device that serves as an interface between an operating system in the host and the shared network adapter.
3. The method of claim 1, further comprising, after identifying the error:
receiving a stop local access network (LAN) command and returning to a start LAN state which purges previously issued control commands.
4. The method of claim 1, wherein, after identifying the error, the data queues are suspended without affecting a current protocol configuration.
5. The method of claim 1, further comprising, after identifying the error:
resetting the control plane; and
reconfiguring the data plane.
6. The method of claim 5, wherein resetting the control plane places the control plane in a state where the only control command that can be executed is a start LAN request.
7. The method of claim 1, wherein the error is an error in the shared network adapter.
8. The method of claim 1, wherein the error is a network interface card/controller (NIC) port transitioning from operational to nonoperational.

9. A system, comprising:
 - one or more processors configured to host a plurality of logical partitions or virtual machines;
 - a shared network adapter configured to provide an interface between the plurality of logical partitions or virtual machines and a network interface card/controller (NIC); and
 - host memory storing computer code configured to perform an operation, the operation comprising:
 - activating control queues between the host and the shared network adapter;
 - issuing control commands to the shared network adapter using the control queues, wherein the control commands configure a data plane for exchanging data between the host and the shared network adapter;
 - activating data queues between the host and the shared network adapter after issuing the control commands; and
 - upon identifying an error, suspending the data queues while permitting the control queues to still transmit control plane data between the host and the shared network adapter.
10. The system of claim 9, wherein the control queues are part of a control plane in a data device that serves as an interface between an operating system in the host and the shared network adapter.
11. The system of claim 9, wherein the operation further comprises, after identifying the error:
 - receiving a stop LAN command and returning to a start LAN state which purges previously issued control commands.
12. The system of claim 9, wherein, after identifying the error, the data queues are suspended without affecting a current protocol configuration.
13. The system of claim 9, wherein the operation further comprises, after identifying the error:
 - resetting the control plane; and
 - reconfiguring the data plane.
14. The system of claim 13, wherein resetting the control plane places the control plane in a state where the only control command that can be executed is a start LAN request.
15. A computer program product, the computer program product comprising:

a computer readable storage medium having computer-readable program code embodied therewith, the computer readable program code executable by one or more computer processors to perform an operation comprises:

activating control queues between a host and a shared network adapter;

issuing control commands to the shared network adapter using the control queues, wherein the control commands configure a data plane for exchanging data between the host and the shared network adapter;

activating data queues between the host and the shared network adapter after issuing the control commands; and

upon identifying an error, suspending the data queues while permitting the control queues to still transmit control plane data between the host and the shared network adapter.

16. The computer program product of claim 15, wherein the control queues are part of a control plane in a data device that serves as an interface between an operating system in the host and the shared network adapter.

17. The computer program product of claim 15, wherein the operation further comprises, after identifying the error:

receiving a stop LAN command and returning to a start LAN state which purges previously issued control commands.

18. The computer program product of claim 15, wherein, after identifying the error, the data queues are suspended without affecting a current protocol configuration.

19. The computer program product of claim 15, wherein the operation further comprises, after identifying the error:

resetting the control plane; and

reconfiguring the data plane.

20. The computer program product of claim 19, wherein resetting the control plane places the control plane in a state where the only control command that can be executed is a start LAN request.