Analysis of Packet Sending and Receiving by Layer 3 Ethernet Switch CPU

Zhao Zhenning
Fan Lihan
Chen Xiao
(Nanjing R&D Institute, ZTE Corporation, Nanjing 210012, China)

Abstract:
Designing an Ethernet switch that can assure normal interaction of protocol packets between switches in a network environment of massive traffic is an important matter. Taking the L3 Ethernet switch based on Application Specific Integrated Circuit (ASIC) as an example, this article analyzes several typical issues about packet receiving and sending by the CPU in a multi-progress environment, including CPU load, software and hardware queue settings, and communication mechanism between CPU and the switch chip. This article gives solutions to these issues mentioned above. The solutions are applicable to Network Processor (NP) issues as well.

In available Layer 3 (L3) Ethernet switch devices, the Layer 2 (L2) switch and L3 routing of message packets are implemented mainly by the switch chip and Network Processor (NP). CPU then is not involved in the switching and routing process. It functions as managing and controlling the switch chip[1].

In such a case, the CPU load is mainly from the regular protocol driver, user configuration driver and external event driver.

Among those drivers, the external event driver is the most random and unpredictable one. The typical external events include the Up/Down state changing of a port, reporting Media Access Control (MAC) address messages (including learning, aging, transferring and more) to upper-layer devices, and CPU receiving and sending packets via Direct Memory Access (DMA).

Among the above listed external events, the processing after the CPU received packets through DMA is the most complicated. When data packets are transferred from the bottom layer to the upper layer software, different protocols will have different processing actions. These actions may be associated with packet sending, port manipulation, multi-table manipulation and more.

Therefore, solutions to the issues about CPU's sending and receiving packets help guarantee the normal interaction of upper layer protocols, and accordingly make the L3 Ethernet switch running stably.

1 Issues about CPU's Sending and Receiving Packets

The analyses of seven issues about packet sending and receiving by the CPU here, are all based on the typical packet sending and receiving mechanism of the CPU: queues are made at the CPU port; packet receiving is fulfilled through DMA; and the ring-type queue is used.

1.1 CPU Load and Packet-receiving Rhythm Control

The packet-receiving rhythm of the CPU is set based on the packet quantity sent to the CPU per unit time. And this packet quantity is determined by the packet processing capability of a switch.

Figure 1 presents two typical processing manners for this issue: CPU fetches packets from DMA at a constant rate; CPU fetches packets from DMA in a burst manner. The upper limit of quantity of packets sent to CPU per unit time is supposed to be *x* per second in the figure.

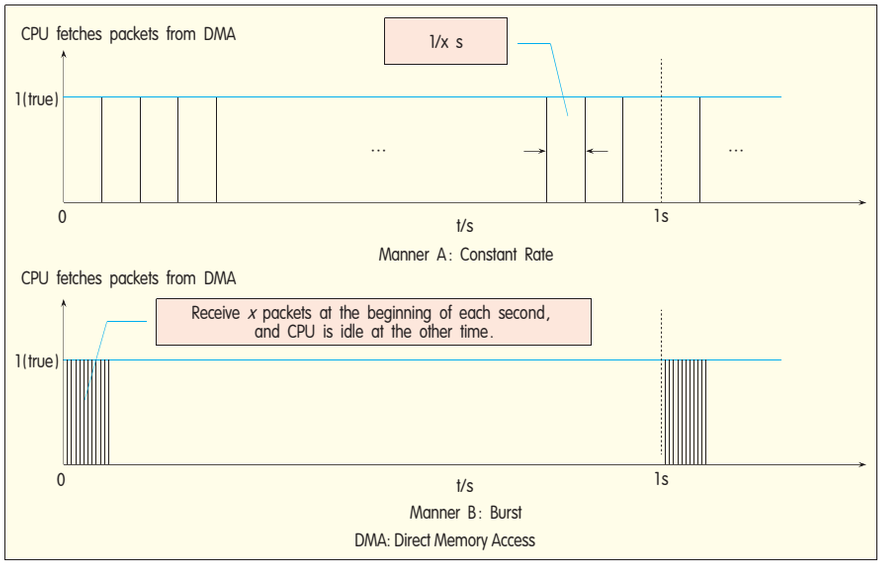
(1) CPU's Fetching Packets at a Constant Rate

When CPU fetches packets from DMA at a constant rate, there is little impact on CPU queue. Moreover, CPU queue is not required to have a high buffering capability; so, it is unnecessary for the queue to be long.

(2) CPU's Fetching Packets in Burst Manner

The hardware receiving queue at one side of the ASIC-based switch chip, together with the ring-type queue in DMA memory, provides certain buffering capability to the switch (against the packets that CPU fetches from the DMA). This buffering capability can be used to extend the control period, to set the control granularity (i.e., the upper limit of packets that CPU fetches in a unit control period), and to use a mechanism, which is similar to degenerative feedback in circuits, of dynamically enable and close the CUP's function of packet-fetching. In this way, the macro control of CPU's packet-fetching rate is realized. Moreover, if the ASIC-based switch chip supports the flow monitoring or shaping functions based on the Token Bucket Algorithm on the egress direction of the CPU[2-3], and if the minimum thresholds of monitoring and shaping satisfy the rate limit of the CPU, this capability can also be used to control the rhythm of the CPU's packet-fetching. This helps reduce the CPU load. As a result, the software

D:\EMAG\2007-09-15/VOL5\DP1.FIT——3PPS/P1



▲ Figure 1. Two ways to control CPU's packet-receiving rate.

processing is much simplified.

1.2 Planning Length of CPU Port Queue

If only the buffering capability of the CPU port is taken into consideration, a longer CPU port queue is preferred. However, the impact of the length on other functions and performances should also be taken into account. For different ASIC chips, the length planning of the CPU port queue varies.

1.3 Zero Copy

During the processing procedure of a data packet, the CPU does not copy the entire packet, but uses pointers as parameters. This way is called zero copy, which greatly improves the processing efficiency of the CPU.

However, the application of zero copy degrades the software processing flexibility to a certain extent. A problem accordingly emerges: if protocol stack requires changing the content of a packet, it will be done at the receiving buffer directly, but if it needs deleting or adding certain fields (such as adding or deleting a layer of tag) in the packet, that is, if the packet length has to be changed, then it is a problem.

Adding or deleting some fields will cause the position change of either a packet header or a packet tailer. If the position of a packet tailer changes, it's required that the total length of the packet does not exceed the buffer boundary; so, the problem is a simple one. However, the field adding and deleting usually occurs near to the header, so the position change of the header is more efficient, and the protocol stack is more inclined to move the position of the header. In this case, the CPU is required to make certain processing when distributing buffer. The process is as follows:

(1) When receiving a packet, the header pointer cannot point to the buffer boundary; rather, it needs to move backwards to allow a certain margin. In addition, a single buffer should have enough space for the Maximum Transmission Unit (MTU) and the margin mentioned.

(2) When releasing a packet, the buffer initial pointer needs to do normalized processing (as shown in Figure 2).

1.4 Interrupt/Polling

Currently, external interrupts of the switch is mainly generated from the switch chip. The major external interrupts include DMA manipulation (such as receiving packets, ending of packet sending and new address message) and some error messages. If interrupt requests are too frequent, the context shift between Interrupt Service Routine (ISR) and other processes will occupy large amount of CPU time. If there are continuously massive interrupt requests, the CPU will be in a busy state. Various protocols don't have sufficient scheduling time and as a result, some serious faults, such as protocol state machine timeout, will occur.

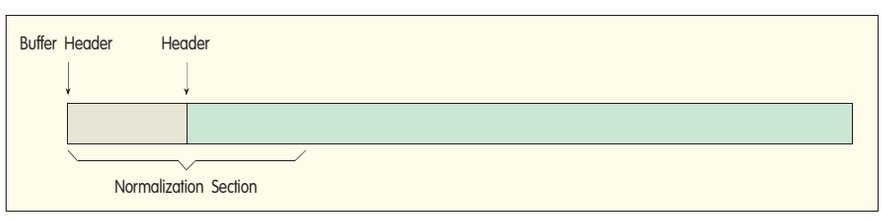
Polling mechanism can be adopted to avoid events trigger with uncontrollable frequency. The regular operation is to use the CPU timer to trigger ISR that is originally triggered by an external interrupt. Since the interval between timer triggers is fixed, the execution frequency of ISR is controlled.

Compared to the external interrupt, polling makes rhythm controllable. The rhythm of external interrupts lies on the frequency of external events; thus, the CPU cannot control it.

However, polling has its own inevitable defect—slow in response, which cannot meet the requirements of some functions that demand higher real-time support. In addition, it is found that when using Ping command to test large packets in L3 interface of the switch, the time delay of the switch applying the polling mode is obviously larger than that of the switch using the interrupt mode.

If a certain mechanism is able to avoid continuous and massive interrupt requests, the CPU will not be too busy, and the real-time processing advantage of the interrupt mode can be reserved as well.

CPU's receiving packets and fetching MAC address message are typical behaviors that will generate massive interrupt events. Taking packet-receiving as an example, the burst mode in the previous part, "CPU Load and Packet-receiving Rhythm Control," controls DMA switch in accordance with real-time traffic, which enables interrupt sources to be controlled. This type of mechanism similar to degenerative feedback is able to avoid continuous interrupt



▲ Figure 2. Normalized processing of a packet.

D:\EMAG\2007-09-15/VOL5\DP1.FIT——3PPS/P2

events sent to the CPU.

Generally, polling is controllable but lacks real-time capability; interrupt is good in real-time capability but difficult in interrupt source control. Therefore, in the initial phase for system designing, specific requirements and the chip's processing manner to external events should be taken into account, to determine which mode should be adopted—interrupt, polling, or both.

1.5 External Event Processing Mechanism in Multi-process Environment

The regular external events (interrupt events) happen in packet receiving and sending of the CPU, including receiving MAC address message and completing MAC table manipulation.

Putting various interrupt events processing in one process factitiously boosts up the coupling of each event and increases mutual restriction of different events.

In order to make the event processing flexible and to reduce the mutual restriction of events in the multi-task operating system, events would initiate their own processes, or they would be classified into several processes in accordance to their processing manners.

1.6 Protocol Packet Protection and CPU Protection

For ASIC-based switch, specific protocol packets require a priority when being sent to the CPU via DMA queue; therefore, certain mechanisms of the ASIC chip are used to assign the protocol packets to designated port queues ensuring their prioritization. This is called protocol packet protection. CPU protection avoids unnecessary data packets taxing the CPU as much as possible.

The two necessary conditions for realizing protocol packet protection are as follows:

(1) The CPU port must support Strict Priority (SP) or Weighted Round-Robin (WRR) scheduling algorithm.

(2) The switch chip needs to have a strong flow classification capability and need to be able to designate different port queues to different flows.

In system scheme design, the following measures can be used to fulfill both protocol packet protection and CPU protection:

(1) Packet-receiving and packet-sending paths should be as clear as possible.

(2) The Access Control List (ACL) of the ASIC chip should be fully used to exactly match different types of protocol packets, and the matching of Layer 4 field is required[4].

Moreover, other functions of the CPU and performance of the switch restrict the implementation of protocol packets protection and CPU protection.

1.7 Avoiding Inefficiency

In multi-task operating system, various events need to be handled within the shortest possible time to ensure that other tasks have adequate chances for scheduling.

So, it is necessary to consider the execution efficiency of any algorithm used. Besides the algorithms themselves, frequent access to certain hardware also takes much time and affects the execution efficiency. However, this is always likely to be neglected in real practice.

2 Conclusions

With the development of Ethernet technology, the processing capability of the switch chip and network processor has been significantly improved; in contrast, the processing performance of the CPU of data switch devices is lagging behind. Moreover, the increasing service types that data switch devices support have higher requirements for service traffic handled by the CPU. In this situation, the limited CPU resource becomes a bottleneck to the development of Ethernet switch. Therefore, it is a prerequisite to the safe and stable operation of the switch equipment that the CPU, the switch chip and the network processor interface conduct good buffering management, queue scheduling and traffic monitoring to ensure proper use of CPU resources. CPU's sending and receiving packets is one of the key subjects of current and future research and development of data switch devices.

References

[1] Chen Xiao, Fan Chengfa, Guo Shihao. CPU Load Analysis of Data Switching Network Equipment [J], Telecommunications Technology, 2004 (8):55-57.
[2] ITEF RFC 2697. A Single Rate Three Color Marker [S]. 1999.
[3] ITEF RFC 2698. A Two Rate Three Color Marker [S]. 1999
[4] Stevens W R. TCP/IP Iillustrated: Vol 1 [M]. Reading, MA, USA:Addison-Wesley,1994: Chapter3,6,11,13,17.

Biographies

Zhao Zhenning graduated from Chongqing University with a major in Communications Engineering. He is a software engineer at Nanjing R&D Institute of ZTE Corporation, and is engaged in the research and development of ZXR10 Ethernet switch.

Fan Lihan received his Master's degree from Central South University with a major in Control Theory and Control Engineering. He is a senior software engineer at Nanjing R&D Institute of ZTE Corporation, and is engaged in the R&D, design and management of high-end Ethernet switch. He participated in several projects funded by the National 863 Program and the National Natural Science Foundation of China, and has been awarded first and third prizes of the Provisional Science and Technology Progress Program of Hunan.

Chen Xiao graduated from Nanjing university of Aeronautics and Astronautics. He is a system engineer at Nanjing R&D Institute of ZTE Corporation. He has participated in the R&D of ZXB10 ATM switch, as well as the R&D, design and management of ZXR10 Ethernet switch. He is now responsible for pre-research of metropolitan Ethernet service switch.

D:\EMAG\2007-09-15/VOL5\DP1.FIT——3PPS/P3