High-Speed Packet Processing Hardware Acceleration Technology: Postprint

Authors: Luo Layong, Xie Yingke, Xie Gaogang

Date: 2017-03-09T00:00:00+00:00

Abstract

The explosive growth in link bandwidth has posed tremendous challenges for high-speed network packet processing. Traditional pure software-based network packet processing can no longer satisfy performance requirements. Current network processors, multi-core chips, and other devices provide hardware acceleration technologies for high-performance network packet processing, offering high-performance implementation approaches for most network applications. For application scenarios with more demanding requirements on performance metrics such as data processing latency, throughput, and packet loss rate, dedicated acceleration hardware is still required. This paper addresses the application requirements for high-performance traffic analysis and control based on Deep Packet Inspection (DPI), and introduces a general high-speed network packet processing hardware acceleration architecture based on Field Programmable Gate Array (FPGA). This architecture accelerates the data acquisition path, achieving line-rate capture of data packets from high-speed links, performs packet forwarding and traffic control through dedicated hardware, optimizes for parallel processing on back-end multi-core servers, and realizes high-performance processing in the control and analysis planes. This paper describes the hardware acceleration methods of this architecture in traffic capture, high-precision clock synchronization, high-speed packet classification, and traffic control. Test results demonstrate that these acceleration methods effectively offload server processing loads and can significantly improve application system performance.

Full Text

Preamble

Vol. 8 No. 6
Information Technology Letters
High-Speed Packet Processing Hardware Acceleration Technologies
Luo Layong, Xie Yingke, Xie Gaogang

Abstract

The dramatic increase in link bandwidth has posed significant challenges for high-speed network packet processing. Traditional pure software-based packet processing can no longer meet performance requirements. Current network processors and multi-core chips provide hardware acceleration technologies for high-performance network packet processing, offering high-performance implementation methods for most network applications. For applications with more stringent performance requirements regarding data processing latency, throughput, and packet loss rate, dedicated acceleration hardware is necessary. This paper addresses the demands of high-performance traffic analysis and control applications based on Deep Packet Inspection (DPI), introducing a general-purpose high-speed network packet processing hardware acceleration architecture based on Field Programmable Gate Arrays (FPGA). This architecture accelerates the data acquisition path, enabling line-rate capture of high-speed link data packets. It employs dedicated hardware for packet forwarding and traffic control, optimizes for parallel processing on back-end multi-core servers, and achieves high-performance processing in both control and analysis planes. This paper presents hardware acceleration methods for traffic acquisition, high-precision clock synchronization, high-speed packet classification, and traffic control within this architecture. Test results demonstrate that these acceleration methods effectively offload server processing loads and significantly improve application system performance.

Keywords: Data acquisition, Load balancing, Clock synchronization, Packet classification, Traffic control

1 Introduction

The exponential growth of network link bandwidth and the diversification of network applications have placed higher demands on both computational and storage resources for network packet processing. The performance growth rate of traditional servers lags far behind that of network bandwidth [1], making it difficult for server performance to meet current high-speed network packet processing requirements. Consequently, network acceleration methods for high-bandwidth and complex network applications have remained a focal point of research in both academia and industry.

Taking intrusion detection and traffic monitoring applications as examples, typical packet processing workflows include traffic acquisition, preprocessing (timestamping, packet classification, filtering, etc.), protocol analysis, and traffic control, as shown in Figure 1 [Figure 1: see original paper]. In this diagram, network data acquisition is performed by ordinary network interface cards (NICs), while complex packet processing tasks such as preprocessing (timestamping, packet classification), traffic statistics, protocol analysis, application decision-making, and traffic control are implemented on back-end servers. In system implementation, I/O performance, main memory performance, and computational perfor-

mance must all be carefully considered. Achieving balanced performance across all processing stages shown in Figure 1 in high-speed network environments faces numerous challenges [2].

Figure 1. Typical packet processing workflow for applications

1. I/O Performance Issues

Network link traffic forms the foundation for all network analysis and research. In high-speed network environments, the performance of network data acquisition directly impacts the overall performance of network traffic processing systems. In earlier times, when network speeds were slower and applications were relatively simple, ordinary NICs combined with data acquisition software could effectively capture and process network traffic. Representative examples include Libpcap [3], BPF [4], and NPF [5]. However, with continuous advancements in network transmission and optical communication technologies, backbone network link rates have reached 10 Gbps. Due to deficiencies in I/O and data transmission paths, traditional network data acquisition methods have become ineffective. Higher network transmission rates impose tremendous demands on the I/O performance and data transmission mechanisms of network traffic processing systems.

2. Main Memory Performance Issues

The diversity of traffic flows requires network traffic processing systems to possess higher-level processing capabilities. However, as processing levels increase, implementing real-time network traffic processing in high-speed environments becomes increasingly difficult, manifesting in three aspects:

- **Main Memory Capacity:** The storage resource requirements of network-layer traffic processing systems are proportional to the number of concurrent flows on the network, while application-layer traffic processing systems' storage requirements are proportional to the number of packet bytes. As network rates continue to increase, the demand for main memory capacity in higher-level network traffic processing becomes substantial and cannot be ignored. Assuming that storing information for one flow requires 200 bytes of space, processing a 10 Gbps network link with 5 million concurrent flows would require 1 GB of main memory space for flow information storage alone. If application-layer traffic processing is considered, requiring storage of payload for each flow, the demand for main memory capacity would increase exponentially.
- **Memory Access Bandwidth:** Network-layer traffic processing is a memory-intensive operation. The number of memory accesses in network-layer processing is proportional to the number of data packets, while application-layer processing memory accesses are proportional to the number of bytes in network link packets. Since each memory access effectively doubles the required memory bandwidth, frequent memory

accesses in high-speed network environments create enormous pressure on main memory bandwidth.

- **Memory Access Latency:** Researchers at UC Berkeley have argued that in computer architecture, performance loss due to memory access latency is more severe than that caused by bandwidth limitations [6]. This problem also exists in network traffic processing systems. Typically, one memory access requires hundreds of CPU clock cycles. However, under extreme conditions in high-speed network environments, the time interval between two packets is far less than the time required for each memory access. For example, on an OC-192 10 Gbps link, the interval between 40-byte minimum-sized packets is only 34 nanoseconds. For a 2 GHz CPU, this interval corresponds to just 68 processor clock cycles, which is clearly insufficient compared to the hundreds of cycles required for memory access, making it impossible to meet line-rate processing requirements for high-speed network traffic.

3. CPU Performance Issues

As described by Gilder's Law [1], network bandwidth continues to grow at a rate of doubling every six months—three times faster than the CPU clock frequency growth rate predicted by Moore' s Law. This widening gap between network link rates and CPU processing capabilities means less CPU time is available for processing each data packet. For instance, when transmitting 64-byte packets, a gigabit Ethernet traffic processing system needs to handle approximately 1.49 million packets per second, whereas an OC-192 10 Gbps link requires processing 18.43 million packets per second—more than ten times higher—while simultaneously achieving higher-level, more complex network traffic processing. Clearly, existing CPU processing capabilities are far from meeting line-rate processing requirements.

To address these performance bottlenecks in purely software-implemented application systems, researchers have begun considering hardware acceleration technologies to accelerate various packet processing stages or offload tasks, thereby reducing CPU processing burden and improving overall application system performance.

This paper first briefly introduces common hardware acceleration platforms, including Network Processors (NP), Field Programmable Gate Arrays (FPGA), Application-Specific Integrated Circuits (ASIC), and embedded multi-core processors, comparing their characteristics. For applications with stringent performance requirements (such as zero packet loss, microsecond-level latency) while requiring certain flexibility, FPGA-based hardware acceleration platforms represent an excellent choice. This paper presents a general-purpose high-speed network packet processing hardware acceleration architecture based on FPGA and discusses research efforts in specific hardware acceleration technologies within this architecture, including traffic acquisition, high-precision clock synchroniza-

tion, high-speed packet classification, and traffic control. Finally, performance evaluations of these acceleration techniques are presented. Test results demonstrate that FPGA-based hardware acceleration methods can effectively offload server processing burden and improve application system performance.

The remainder of this paper is organized as follows: Section 2 introduces common hardware acceleration technologies and analyzes the advantages and disadvantages of various hardware platforms. Section 3 presents an FPGA-based hardware acceleration solution, proposing a general-purpose high-speed packet processing hardware acceleration architecture for online traffic analysis and control applications. Section 4 describes specific acceleration methods based on this architecture, including traffic acquisition, high-precision time synchronization, high-speed packet classification, and traffic control. Section 5 evaluates the performance of related acceleration techniques. Finally, Section 6 provides a brief summary of the research work.

2 High-Speed Packet Processing Hardware Acceleration Technologies

To address performance bottlenecks in purely software-implemented application systems for high-speed network packet processing, academia and industry have continuously researched hardware acceleration technologies for high-speed packet processing, resulting in several network processing hardware acceleration platforms.

Common hardware acceleration platforms include network processors, embedded multi-core processors, field programmable gate arrays, and application-specific integrated circuits.

Network processors are a class of high-performance programmable specialized instruction set processors. They typically contain multiple packet processing engines with computing and storage capabilities that work in parallel to achieve high-speed, complex network data processing. Network processors offer high processing performance, with representative products including Intel' s IXP series [7] and IBM' s PowerNP series [8]. However, network processors use microcode programming, which is complex and suffers from poor code reusability and portability, leading to decreasing usage in practical systems.

In recent years, rapidly advancing multi-core technology has changed industry trends, providing new opportunities for parallel processing of network algorithms and applications. Multi-core platforms launched by several companies have gradually become hot topics. Typical representatives include Cavium' s [9] OCTEON II series and Tilera' s [10] TILE64 and TILE-Gx series multi-core processors. These embedded multi-core processors are designed for specific applications (such as digital multimedia and network packet processing) and thus integrate dedicated hardware acceleration engines and specialized interfaces. For network packet processing requirements, such processors typically integrate rich network interfaces (e.g., SGMII, XAUI, Interlaken) to meet various high-speed

link data acquisition needs; high-speed memory interfaces (e.g., DDR3) to satisfy storage bandwidth requirements for high-speed network data processing; high-speed buses (e.g., PCIe 2.0) to resolve I/O bottlenecks when communicating with hosts; dedicated hardware acceleration engines such as Crypto Security for network security applications and HFA engines for deep packet inspection (used for pattern matching); and a large number of processing cores (e.g., 32, 64, or 100 cores) providing powerful parallel processing capabilities that address traditional CPU processing insufficiencies. C/C++-based programming greatly enhances generality, flexibility, and portability. However, new problems introduced by multi-core architectures, such as inter-core communication latency and data sharing overhead, may become performance bottlenecks in applications requiring low forwarding latency and high throughput. Additionally, for certain network applications with limited parallelism, the advantages of multi-core parallel processing cannot be realized.

Since dedicated hardware logic can achieve the highest performance, for specific operations such as encryption/decryption and string matching, high-performance application systems still require some ASICs. However, ASIC development involves large investments, long development cycles, and difficult upgrades, making them suitable only for specific applications. Another hardware acceleration technology based on FPGA has gained widespread application. FPGAs inherently possess high-speed, parallel, and programmable characteristics, making them highly suitable for high-speed network packet processing hardware acceleration. Currently, FPGAs have seen significant developments in capacity and speed, enabling implementation of large-scale complex logic. Network hardware acceleration solutions based on this platform have received extensive research attention [11]. Representative research and products include the University of Washington ARL Laboratory' s FPX platform [12], Stanford University' s NetFPGA project [13], and Endace' s DAG series monitoring cards [14].

Table 1 summarizes the aforementioned hardware platforms.

Table 1. Comparison of Common High-Speed Network Packet Processing Hardware Acceleration Platforms

Platform	Typical Application Domain	Advantages	Disadvantages	Current Status
Network Processor	High-speed packet processing	Integrates multiple parallel processing engines with strong processing capability	Microcode programming is difficult; poor code reusability and portability	Traditional microcode-programmed network processors are rarely used today, gradually being replaced by FPGA and multi-core platforms
FPGA	High-speed, parallel, programmable applications	High speed, parallel, programmable	Limited flexibility; difficult to implement complex packet processing	Widely used in research and specific applications
ASIC	Specific high-performance domains	Highest speed; can significantly reduce costs in mass production	Long development cycles; difficult upgrades; unsuitable for research platforms	Mature dedicated network acceleration products

Platform	Typical Application Domain	Advantages	Disadvantages	Current Status
Embedded Multi-core	Network packet processing with high parallelism	High parallelism; integrated network packet processing acceleration engines; rich network interfaces; high-speed bus interfaces	Expensive; parallel programming and debugging difficulties; new issues like inter-core communication latency and data sharing overhead	Suitable for network packet processing applications with significant parallelism to exploit

Overall, these network packet processing technologies focus on improving I/O performance, main memory performance, and computational performance. For network applications in high-speed link environments, distributed parallel or pipelined processing methods must be adopted, with hardware acceleration components offloading critical computations. This paper discusses relevant hardware acceleration technologies for online traffic analysis and control applications. Such applications demand high packet processing performance (e.g., acquisition performance, forwarding latency) while requiring complex control plane processing. Therefore, FPGAs are typically selected as the hardware acceleration platform to implement line-rate processing in the data plane, while high-performance servers handle complex protocol analysis and policy control. In hardware platform design, the core challenge is balancing I/O performance, main memory performance, and computational performance to optimize the entire processing workflow, fully exploit system computing capabilities, and achieve overall system performance optimization.

3 High-Speed Packet Processing Hardware Acceleration Architecture

To accelerate traditional pure software packet processing workflows, our architecture design follows these principles:

1. **Ensure line-rate acquisition:** Use FPGA-based acceleration cards to replace ordinary NICs, optimizing the data acquisition path to achieve line-rate capture of high-speed traffic.

2. **Fully exploit multi-core parallelism:** Optimize hardware for back-end multi-core server parallel processing by utilizing multi-channel high-speed DMA (Direct Memory Access) engines and load balancing technology on the acceleration card to evenly distribute traffic to multiple back-end cores, directly supporting data parallelism from the hardware level and fully leveraging the parallel processing capabilities of back-end multi-core processors.
3. **Fully offload software processing burden:** Leverage the inherent parallel and high-speed characteristics of FPGAs to implement function modules that are time-consuming in software but can be easily processed at high speed in hardware, thereby fully offloading back-end server processing burdens. Examples include hardware-implemented high-precision timestamps, high-speed packet classification, and high-precision traffic control units.

Based on these considerations and targeting typical applications such as online traffic analysis and control, we designed the high-speed packet processing hardware acceleration architecture shown in Figure 2 [Figure 2: see original paper]. The architecture consists of a control plane and a data plane: the data plane completes various packet processing tasks, including protocol preprocessing, application acceleration, data acquisition, and forwarding; the control plane manages and configures the system and provides open system control interfaces to users.

The data plane structure fully employs pipelining and parallelization techniques, as shown in Figure 2. Each packet is immediately stamped with a hardware timestamp upon entering the receive queue. The preprocessing stage first parses and preprocesses the packet' s Layer 3 and Layer 4 protocols, then selectively directs packets into one or multiple functional units—such as the packet classification unit or flow table lookup unit—based on different application requirements through programmable interface configurations. All processing units execute in parallel, with their results arbitrated by an arbitration module. The arbitration module is also responsible for encapsulating packets, packaging the processing results from front-end units along with the entire packet into a custom-format record for subsequent processing. At the back end of the pipeline, packets are duplicated into two paths: one path enters the acquisition component and is transmitted to the back-end server for further processing via load balancing and multi-channel high-speed DMA engines; the other path enters the forwarding engine, providing a fast packet forwarding path. Based on specific application requirements, traffic can be selectively cleaned and controlled.

Key technologies to be implemented in the system design include:

- In multi-core, multi-processor processing environments, distributing network data flows to corresponding processing threads through hardware —i.e., implementing high-performance packet classification in hardware. High-performance packet classification algorithms and their implementation are critical problems that require focused solutions.

- Implementing high-performance data acquisition. To ensure packet loss prevention, research is needed on multi-buffer queues, DMA technology, memory circular buffering techniques, and load balancing technology to guarantee high-performance data acquisition from both physical channels and processing mechanisms.
- Precise traffic control technology. For online application systems, implementing traffic control in hardware can effectively reduce latency. Developing scalable, high-precision traffic control units is essential work to ensure system usability.
- Implementation of relevant hardware acceleration units. For example, network analysis requires precise analysis of packet time series relationships, demanding high-precision timestamps. In OC-192 links, the minimum packet interval is only 30.5 ns [15], requiring timestamp resolution at the nanosecond level. Generating timestamps in hardware ensures both high precision and effectively reduces software computational load.

4 Hardware Acceleration Methods for High-Speed Packet Processing

Based on the acceleration architecture proposed in the previous section, our research group has conducted a series of studies in data acquisition, clock synchronization technology, packet classification technology, and traffic control technology. These aspects are introduced below.

4.1 Parallel-Optimized Line-Rate Data Acquisition Structure

Network traffic acquisition forms the foundation for comprehensive statistical analysis of network traffic. Traditional traffic acquisition is typically software-based, using ordinary NICs combined with general-purpose CPUs to complete network traffic acquisition and analysis [16-17]. This approach offers simplicity and good portability. However, rough estimates [18] indicate that processing 1 bit of network data consumes 1 Hz of CPU processing capability. As link bandwidth increases, traditional acquisition methods encounter bottlenecks in interrupt handling and memory copying, making it impossible to guarantee packet loss-free capture in high-speed links. With continuous upgrades in FPGA capacity and speed, hardware acceleration solutions based on this platform have received extensive research attention [11]. Representative research and products include the University of Washington ARL Laboratory' s FPX platform [12] and Endace' s DAG series monitoring cards [19]. Endace' s high-speed traffic acquisition card (DAG card) based on FPGA can achieve line-rate acquisition in 1G, 2.5G, and 10G link environments, occupying almost no CPU resources during packet capture and dedicating all processing capability to applications.

To further improve data acquisition performance and optimize for parallel processing on back-end multi-core servers, we propose a parallel-optimized line-rate data acquisition structure based on FPGA, as shown in Figure 3 [Figure 3: see

original paper].

Figure 3. Parallel-optimized acquisition structure

In traditional acquisition structures, the DMA (Direct Memory Access) engine requires only a single transmission channel. To offload software multi-threading overhead from shared data locking, we designed an 8-channel independent high-speed PCIe DMA engine based on FPGA. Using a load balancing unit to implement data distribution in hardware, packets are transmitted through eight DMA channels to different memory buffers on the host. Consequently, packets entering the back-end server have already achieved data-level parallel decomposition, with completely independent data for each back-end processing thread, enabling lock-free programming and avoiding thread mutual exclusion overhead. Simultaneously, the back-end host can employ circular buffer technology and zero-copy techniques [20] to avoid data copying overhead.

In DMA engine design, to better accommodate high-speed link data acquisition requirements, our DMA engine supports polling mechanism control, thereby avoiding interrupt overhead.

Processing capabilities of back-end multi-core parallel processing units may vary. If traffic is evenly distributed without considering these capability differences, it may result in frequently idle high-capacity units while low-capacity units cannot process all packets in time. Therefore, we propose a Feedback-based Dynamic Load Balance algorithm (FDLB) [21-22] that dynamically adjusts load distribution according to processing unit capabilities while maintaining session integrity as much as possible. When input traffic exceeds back-end processing capacity, the load balancing unit introduces Random Early Detection (RED) [23] to drop packets early and avoid congestion.

4.2 High-Precision Clock Synchronization Technology [21]

Network applications such as traffic engineering and quality of service all require time information [24-27]. The precision of clock information directly affects the accuracy of application system computation results. High-precision clock synchronization involves two problems [21]:

1. **Multi-system clock synchronization:** Clocks of multiple nodes in distributed network application systems must maintain precise synchronization.

Among pure software time synchronization methods, the Network Time Protocol (NTP) is most widely used [21]. Nodes use the NTP protocol to synchronize with time servers in the network, estimating round-trip delay of packets to calculate clock offset and complete synchronization. NTP can achieve millisecond-level synchronization precision [28]. Literature [29] achieved stable clock synchronization with errors within 1 s using NTP on a real-time Linux operating system. Software clock synchronization methods are simple to implement but suffer from low precision and unstable performance. Horauer et al. proposed an

Ethernet-based clock synchronization technology called SynUTC [30] that uses hardware-generated timestamps with synchronization errors of 100 ns. Literature [31] designed a dedicated packet capture DAG card that uses hardware to timestamp all captured packets, with all cards synchronized to GPS clocks and timestamp deviations within 100 ns. However, the requirement for GPS makes the system cost prohibitively high.

To meet clock synchronization requirements for numerous network applications, our research group proposed a software-hardware combined method to improve system clock synchronization precision [21]. This method uses hardware-generated timestamps to eliminate various software delay effects. Additionally, we designed a Prediction-based Clock Synchronization (PCS) algorithm to complete clock synchronization across nodes. The designed clock synchronization subsystem offers considerable flexibility: it can use GPS clocks as the clock source to synchronize nodes, or alternatively, achieve synchronization across multiple nodes using serial communication without GPS clock sources, with synchronization precision no lower than GPS-based approaches.

The hardware timestamp generation circuit is shown in Figure 4 [Figure 4: see original paper] [21].

Figure 4. Timestamp generation circuit

Clock synchronization is essentially the process of adjusting a system' s clock frequency to match a reference clock frequency. In the timestamp generation circuit, this function is performed by a Direct Digital Synthesizer (DDS). By adjusting the DDS frequency control word, clocks of various frequencies can be obtained. The principle of the PCS algorithm [21] is described in Figure 5 [Figure 5: see original paper]. The algorithm consists of two phases: startup and synchronization. During the startup phase, slave nodes complete correction of fractional second deviations in timestamps. During the synchronization phase, slave nodes fine-tune the current DDS frequency control word based on previous results to synchronize their clocks with the master node.

Figure 5. PCS algorithm schematic diagram

4.3 General Packet Classification Structure RSTCAM [21]

Packet classification technology supports the categorization of arriving packets into different data flows in network devices. It uses different rules to identify various data flows, with each rule specifying the corresponding action to be executed for packets in that flow based on analysis of packet header fields. Typical packet classification algorithms mainly fall into three categories [21]: decision tree-based algorithms, decomposition-parallel algorithms, and TCAM-based algorithms. Decision tree-based algorithms improve upon linear search by using tree structures instead of traditional linear linked lists. A decision tree is first built based on the classification rule set, and packets traverse the tree during

classification to obtain results. Typical representatives include Grid-of-Tries [32], FIS-tree [33], AQT [34], EGT-PC [35], HiCuts [36], and HyperCuts [37]. Decomposition-parallel algorithms parallelize the classification problem by decomposing multi-dimensional packet classification into multiple one-dimensional classification problems (e.g., BV [38], ABV [39]) or dividing a large classification space into multiple subspaces (e.g., Tuple Space [40]), reducing algorithmic complexity and improving performance. TCAM-based algorithms introduce hardware solutions. Due to TCAM' s extremely fast classification speed, commercial packet classifiers currently adopt TCAM-based approaches. However, the massive parallel structure in TCAMs leads to high power consumption and many limitations in storage density and scalability. Therefore, research in this area focuses on addressing these issues, with representative works including ETCAM [41], EaseCAM [42], and CoolCAMs [43].

For common five-tuple-based packet classification algorithms, since the five fields in the tuple are independent, each field can be matched first, and the results merged to obtain the final match. Literature [38] first proposed this idea and implemented a packet classifier using five 128Kb SRAM chips. This classifier supports 512 rules and can perform 1 million lookups per second on average. Based on similar principles, our RSTCAM [21] design uses five TCAMs, each storing one field of the five-tuple, with final classification results obtained by "ANDing" the matching results from each field.

The RSTCAM structure is shown in Figure 6 [Figure 6: see original paper] [21].

Figure 6. RSTCAM structure

RSTCAM converts the original five-dimensional lookup into five one-dimensional lookups, offering the following advantages [21]:

1. **Increased operating frequency:** TCAM operating frequency is related to its bit width—the wider the bit width, the lower the frequency. By converting five-dimensional lookup into five one-dimensional lookups, the bit width of each TCAM is reduced, thereby increasing operating frequency.
2. **Reduced resource consumption:** In classification rule sets, many rules often share identical fields. By merging identical fields, the capacity of each TCAM can be significantly reduced.
3. **Reduced system power consumption:** TCAM power consumption is primarily generated by its internal parallel structure. The RSTCAM structure reduces the bit width of TCAM parallel structures, thereby lowering power consumption. Additionally, since the five TCAMs are independent, individual TCAMs can be powered on or off, enabling a trade-off between classification precision and power consumption.
4. **Easier range matching implementation:** Fields expressed as ranges in classification rules typically require conversion to prefix representation. In the worst case, a w-bit range requires conversion to 2(1)−w prefixes. For a rule iR containing d fields, each expressed as ranges with bit widths w1, w2, ···, wd, the worst-case scenario requires conversion to

$(2^{\wedge}(w1+1)-2) \times (2^{\wedge}(w2+1)-2) \times \cdots \times (2^{\wedge}(wd+1)-2)$ prefix rules. Using the RSTCAM structure, since fields are independent, only $(2^{\wedge}(w1+1)-2) + (2^{\wedge}(w2+1)-2) + \cdots + (2^{\wedge}(wd+1)-2)$ rules are needed in the worst case.

4.4 High-Precision Scalable Traffic Control Technology

Traffic control is a key technology for network monitoring, network security, and quality of service assurance. Ren Fengyuan et al. [44] divided traffic control research into two stages based on historical evolution: end-to-end traffic control (hosts) and intermediate node traffic control (routers). Early traffic control primarily utilized the sliding window mechanism in TCP to implement control at end systems. Later, researchers conducted in-depth studies to continuously improve algorithm defects, resulting in five major versions of TCP traffic control algorithms: Tahoe [45], Reno [46], NewReno [47], SACK [48], and Vegas [49]. With technological development, researchers realized that implementing traffic control only at end systems could hardly meet quality of service requirements. Consequently, extensive research began focusing on traffic control at network intermediate node devices (routers). Router traffic control research concentrates on queue management and queue scheduling. Queue management algorithms study when and which packets to drop to maintain small queue lengths, while queue scheduling algorithms determine the transmission order of packets from multiple waiting queues and bandwidth allocation among queues. Typical queue management algorithms include RED [23] and various RED-based improvements such as ARED (Adaptive RED) [50] and Balanced-RED [51]. Representative queue scheduling research includes GPS (Generalized Processor Sharing) model [52], WRR (Weighted Round Robin) [53], and WFQ (Weighted Fair Queue) [54].

Since router traffic control is based on queue management and scheduling, and queue resources are limited, it is difficult to meet the needs for fine-grained bandwidth management at the service flow level. To address this issue, this paper proposes the concept of "virtual channels."

Virtual channels are identified by Virtual Channel IDs. Each packet carries a virtual channel ID and selects which virtual channel to enter based on this ID. At each virtual channel entrance, a control switch is designed, as shown in Figure 7 [Figure 7: see original paper].

Figure 7. Virtual channel schematic diagram

In Figure 7, the control switch at each virtual channel entrance is essentially a packet admission check unit. This unit examines each packet arriving at the virtual channel entrance and decides whether to allow it into the channel. If not permitted, the packet is dropped. The unit includes several key parameters:

- **T:** Basic time unit. The timeline is divided into multiple time units T, with traffic control implemented within each time unit.

- **B:** Counter for the total length of data packets entering the virtual channel within time unit T. At the start of each time unit T, B is initialized to 0.
- **W:** Threshold for total packet length. Within each time unit T, the total byte count B of packets entering the virtual channel cannot exceed W.

Let Li be the length of the ith packet entering during time unit T. The condition for the ith packet to be admitted is: B + Li ≤ W. If this condition is not met, the ith packet is dropped. If the ith packet successfully enters the virtual channel, then B = B + Li; otherwise, B remains unchanged.

For all time units T, if parameter W remains constant, we can ensure the virtual channel bandwidth R satisfies: R = W/T. This represents the traffic limiting scenario.

If we modify the threshold Wm for the mth time unit based on the final values of Wm-1 and Bm-1 from the (m-1)th time unit as follows: Wm = Wm-1 + (W - Bm-1), then the traffic rate R of this virtual channel can satisfy: R = W/T. This represents precise traffic control. However, if we continuously adjust according to this formula, when actual traffic is less than the control value, Wm will increase without limit. Eventually, when actual traffic exceeds the control value, traffic will remain uncontrollable for a long period. Therefore, when Wm exceeds a certain threshold, its adjustment component must be cleared, resetting Wm = W.

Virtual channels themselves do not consume any actual logic resources; they are merely an abstraction of data paths for various service flows when controlling multiple flows simultaneously. The actual logic resources are consumed by the control switches at each virtual channel entrance. In an FPGA, implementing 1024 virtual channels requires 1024 control switches with different parameters. However, regardless of how many concurrent flows exist in the current link, packets are transmitted and processed serially at the micro level. Therefore, it is not necessary to design 1024 different control switches; instead, a single control unit storing control parameters (T, B, W, etc.) for 1024 different virtual channels should be implemented. For each packet entering the control unit, the corresponding virtual channel' s control parameters are read based on its carried virtual channel ID, and admission checking is performed. After processing a packet, the modified parameter B is stored, preparing for the next packet.

Additionally, the time unit parameter T can be shared globally across all virtual channels. Therefore, to implement 1024 different traffic control strategies, 1024 virtual channels are needed, consuming logic resources including: one unified control unit, storage for 1024 sets of parameters B and W, and management of one global parameter T. Thus, adding one control channel only requires adding storage resources for two parameters, demonstrating excellent scalability of this method with respect to the number of traffic control channels.

Finally, a large output buffer queue is needed to shape all flows uniformly. Since packet transmission is serial at the micro level, when network traffic from multiple virtual channels converges at the output queue, no scheduling is required.

5 Performance Evaluation

We implemented multiple prototype systems for packet processing hardware acceleration based on Xilinx's V5 series FPGA to meet requirements for different network links such as GE (Gigabit Ethernet), 10GE, and 10G POS (Packet Over Sonet). Using Spirent's TestCenter network tester, we conducted comprehensive performance tests on the system's acceleration capabilities, primarily evaluating the following aspects:

1. Line-Rate Acquisition Test [2]

Figure 8 [Figure 8: see original paper] shows packet rates for data acquisition under test traffic with various packet lengths. Except for the most extreme case (64-byte packets), the acceleration platform can achieve line-rate acquisition of 10 Gbps network traffic. In the extreme case of 64-byte packets, a small amount of packet loss occurs. To reduce back-end CPU processing pressure, each transmitted packet includes 16 bytes of additional information, occupying PCI-Express bus bandwidth and making the actual bus transmission bandwidth higher than the test instrument' s sending bandwidth. When four CPUs simultaneously perform data acquisition, based on test results (maximum captured packet rate), the PCI-Express bus effective payload bandwidth reaches 11.2 Gbps, approaching the theoretical limit and causing packet loss.

Figure 8. Data acquisition test

2. Dual-Card Clock Synchronization Precision Test [21]

Two acceleration cards were installed in different servers and interconnected via RS422. First, Network Time Protocol (NTP) was run on both servers to achieve second-level synchronization. Then the PCS synchronization algorithm was started with master and slave modes configured. The measured timestamp deviation between master and slave nodes is shown in Figure 9 [Figure 9: see original paper]. Due to space limitations, the left half of Figure 9 only shows a graph of 6,956 points over 2 hours, while the right half shows statistics from 12 hours of measurement. The results demonstrate that the PCS algorithm successfully synchronized the two cards by adjusting frequency control words, with most timestamp deviations controlled within 0-100 ns. This achieves precision comparable to GPS-based synchronization.

Figure 9. Timestamp deviation between master and slave nodes and deviation distribution

3. Packet Classification Function Test [21]

To verify packet classification functionality, we configured the classifier with: Rule 1 [X, X, UDP, X, X] matching all UDP traffic; Rule 10 [192.168.0.X, X, TCP, X, X] matching all TCP traffic with source IP 192.168.0.X; Rule 50 [192.168.1.X, 192.168.100.X, TCP, X, X] matching all TCP traffic with

source/destination IPs 192.168.1.X and 192.168.100.X; Rule 150 [192.168.1.X, 192.168.200.X, TCP, 1-255, X] matching all TCP traffic with source/destination IPs 192.168.1.X and 192.168.200.X and source ports 1-255; the final rule matching all traffic; and all other rules set to zero.

The test instrument generated five traffic types: stream0 as UDP traffic; stream1 as TCP traffic with source IP 192.168.0.X; stream2 as TCP traffic with source/destination IPs 192.168.1.X and 192.168.100.X; stream3 as TCP traffic with source/destination IPs 192.168.1.X and 192.168.200.X and source ports 1-255; stream4 as TCP traffic with source/destination IPs 192.168.1.X and 192.168.200.X and source ports 1024-4095. All test traffic used 64-byte small packets, with each type occupying 20% of total bandwidth. During testing, traffic matching rules 1, 10, 50, and 150 was gradually filtered, with measured bandwidth for each stream shown in Figure 10 [Figure 10: see original paper].

At step 0, all streams had equal bandwidth (157,446,808 bps). At step 1, traffic matching rule 1 was filtered (by modifying the operation code in the lookup table corresponding to classification rule 1 to 0xf to filter matching packets), causing stream0 bandwidth to become 0 while other streams increased to 184,325,000 bps (no packet loss except for filtered packets). At step 2, traffic matching rule 2 was filtered, making stream1 bandwidth 0. By step 4, only stream4 traffic remained. The test results in Figure 10 demonstrate that the RSTCAM packet classification engine can correctly classify packets at 10 Gbps rates.

Figure 10. Packet classification function test

4. Traffic Control Precision Test

Using TestCenter to generate random-length traffic at 100% rate, we controlled traffic through our acceleration card' s traffic control function at rates of 100 Kbps, 1 Mbps, 10 Mbps, 50 Mbps, 100 Mbps, and 200 Mbps. Each test lasted 30 minutes, sampling one rate value per second. Traffic control precision test results are shown in Figure 11 [Figure 11: see original paper].

Figure 11. Traffic control precision test

Test results indicate that larger traffic control values yield smaller relative errors. Errors are approximately 0.1% for small traffic rates and below 0.1% on average for large traffic rates.

6 Summary and Outlook

The massive volume of high-speed network data and more complex processing demands pose enormous challenges for high-speed network packet processing. Traditional server performance falls far short of current high-speed packet processing requirements, with performance bottlenecks existing in many stages. To comprehensively improve high-speed network packet processing performance, we propose a general-purpose FPGA-based high-speed network packet processing acceleration architecture with three key characteristics: (1) high-speed traffic

line-rate acquisition; (2) optimization for multi-core parallel processing; (3) hardware fully offloading software burden. Based on this architecture, our research group has proposed specific hardware acceleration methods in traffic acquisition technology, high-precision clock synchronization technology, high-speed packet classification technology, and traffic control technology. Test results demonstrate that our proposed hardware acceleration methods are effective and can significantly improve overall application system performance.

Based on our research results, further in-depth investigation can be conducted in the following areas:

1. Research on key technologies for higher-speed link (e.g., 40 Gbps, 100 Gbps) packet processing to meet the continuously upgrading backbone network bandwidth demands.
2. Research on hardware acceleration technology for application-layer deep packet inspection. Deep packet inspection is a key technology for protocol analysis and network security applications, but traditional software algorithms struggle to meet performance requirements for high-speed link deep packet inspection. Therefore, hardware acceleration technology for higher rates and larger rule sets represents a worthwhile research direction.
3. High-speed packet processing acceleration technology based on multi-core platforms. The inherent parallel, high-speed, and programmable characteristics of FPGAs make them highly suitable for hardware acceleration of high-speed packet processing. However, FPGAs have limited flexibility and generality, making it difficult to meet complex network processing requirements. The rise of multi-core (Multi-Core) and many-core (Many-Core) technologies provides new solutions for high-speed packet processing. How to fully exploit packet processing parallelism and leverage the advantages of multi-CPU parallel processing is a direction worthy of in-depth research.

References

[1] G. Gilder, "Fiber Keeps Its Promise: Get ready. Bandwidth will triple each year for the next 25," Forbes (Apr), 1997.

[2] Zhu Chao, "Research on Optimization Methods for High-Speed Network Traffic Processing Systems," Ph.D. dissertation, Graduate University of Chinese Academy of Sciences.

[3] V. Jacobson, et al. Libpcap. Available: http://www.tcpdump.org

[4] M. Steven and J. Van, "The BSD packet filter: a new architecture for user-level packet capture," presented at the Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings, San Diego, California, 1993.

[5] F. Risso and L. Degioanni, "An Architecture for High Performance Network Analysis," presented at the Proceedings of the Sixth IEEE Symposium on

Computers and Communications, 2001.

[6] K. Asanovic, et al., "The Landscape of Parallel Computing Research: A View from Berkeley," Electrical Engineering and Computer Sciences, University of California at Berkeley, 2006.

[7] N. Shah, et al., "NP-Click: A programming model for the Intel IXP1200," in 2nd Workshop on Network Processors (NP-2) at the 9th International Symposium on High Performance Computer Architecture (HPCA-9), Anaheim, CA, ed, 2003.

[8] J. Allen Jr, et al., "IBM PowerNP network processor: Hardware, software, and applications," IBM Journal of Research and Development, vol. 47, pp. 177-177, 2003.

[9] Cavium. OCTEON Multi-Core Processor. Available: http://www.caviumnetworks.com/

[10] Tilera. Multicore Processors. Available: http://www.tilera.com/

[11] G. Sun and Z. S. He, "A Real-Time Multi-Channel Signal Acquisition Card Based on PCI Express Interface," Proceedings of the International Conference on Communication Software and Networks, pp. 20-24 848, 2009.

[12] J. W. Lockwood, et al., "Reprogrammable network packet processing on the field programmable port extender (FPX)," in 9th International Symposium on Field Programmable Gate Arrays(FPGA), Monterey, California, United States, 2001, pp. 87-93.

[13] J. Lockwood, et al., "NetFPGA–an open platform for gigabit-rate network switching and routing," 2007, pp. 160-161.

[14] S. Donnelly and P. Limited, "Dag packet capture performance," White Paper, August, 2006.

[15] S. F. Donnelly, "High Precision Timing in Passive Measurements of Data Networks," Ph D, University of Waikato, 2002.

[16] V. Y. Hnatyshin and A. F. Lobo, "Undergraduate data communications and networking projects using opnet and wireshark software," presented at the Proceedings of the 39th SIGCSE technical symposium on Computer science education, Portland, OR, USA, 2008.

[17] L. Deri, "nProbe: an Open Source NetFlow Probe for Gigabit Networks," in TERENA Networking Conference, 2003.

[18] E. Yeh, et al., "Introduction to TCP/IP offload engine (TOE)," 10 Gigabit Ethernet Allience Version, vol. 1, 2002.

[19] Endace. (2010, DAG Network Monitor Cards. Available: http://www.endace.com/

[20] Zhao Zili, "Research on Performance Optimization Technology for 10G Network Traffic Processing Systems," Master' s thesis, Graduate University of Chinese Academy of Sciences, 2009.

[21] Wang Jiandong, "Research on Key Technologies for 10G Backbone Network Data Acquisition and Preprocessing," Ph.D. dissertation, Graduate University of Chinese Academy of Sciences, 2009.

[22] Wang Jiandong, Zhu Chao, Xie Yingke, Han Chengde, Zhao Zili, "Research on FPGA-based 10G Traffic Parallel Real-Time Processing System," Computer Research and Development, vol. 46, pp. 177-185, 2009.

[23] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on networking, vol. 1, pp. 397-413, 1993.

[24] B. B. K. Salamatian, and T. Bugnazet, "Cross traffic estimation by loss process analysis," presented at the In Proceedings of ITC Specialist Seminar on Internet Traffic Engineering and Traffic Management, Wurzburg, Germany, 2003.

[25] S. Joel, et al., "Improving accuracy in end-to-end packet loss measurement," presented at the Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, Philadelphia, Pennsylvania, USA, 2005.

[26] P. Vern, "Strategies for sound internet measurement," presented at the Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Taormina, Sicily, Italy, 2004.

[27] M. T. Y. Kitatsujji, S. Katsuno, and Y. Oie, "Usefulness of precise time-stamping for exposing network characteristics on high-speed links," presented at the In Proc. of SPIE ITcom, Philadelphia, USA, 2004.

[28] A. T. D. Mills, and B.C. Huffman, "Internet timekeeping around the globe," presented at the In Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting, Long Beach, CA, 1997.

[29] P. Attila, et al., "PC based precision timing without GPS," presented at the Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Marina Del Rey, California, 2002.

[30] M. H. R. Hoeller, G. Griedling, N. Keroe, U. Schmid, K. Schossmaier, "SynUTC –High Precision Time Synchronization over Ethernet Networks," presented at the In Proceedings of the 8th Workshop on Electronics for LHC Experiments, Colmar, France, 2002.

[31] J. Micheel, et al., "Precision timestamping of network packets," Imw 2001: Proceedings of the First Acm Sigcomm Internet Measurement Workshop, pp. 273-277 311, 2001.

[32] V. Srinivasan, et al., "Fast and scalable layer four switching," presented at the Proceedings of the ACM SIGCOMM ' 98 conference on Applications, technologies, architectures, and protocols for computer communication, Vancouver, British Columbia, Canada, 1998.

[33] A. Feldman and S. Muthukrishnan, "Tradeoffs for packet classification," in INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 2000, pp. 1193-1202 vol.3.

[34] M. B. Milind, et al., "Space Decomposition Techniques for Fast Layer-4 Switching," presented at the Proceedings of the IFIP TC6 WG6.1 & WG6.4 / IEEE ComSoc TC on Gigabit Networking Sixth International Workshop on Protocols for High Speed Networks VI, 2000.

[35] F. Baboescu, et al., "Packet classification for core routers: is there an alternative to CAMs?,"in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, 2003, pp. 53-63 vol.1.

[36] P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings," IEEE Micro, vol. 20, p. 34~41, 2000.

[37] S. Sumeet, et al., "Packet classification using multidimensional cutting," presented at the Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, Karlsruhe, Germany, 2003.

[38] T. V. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," presented at the Proceedings of the ACM SIGCOMM ' 98 conference on Applications, technologies, architectures, and protocols for computer communication, Vancouver, British Columbia, Canada, 1998.

[39] B. Florin and V. George, "Scalable packet classification," vol. 13, ed: IEEE Press, 2005, pp. 1600-1615.

[40] V. Srinivasan, et al., "Packet classification using tuple space search," presented at the Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication, Cambridge, Massachusetts, United States, 1999.

[41] S. Ed, et al., "Packet Classification Using Extended TCAMs," presented at the Proceedings of the 11th IEEE International Conference on Network Protocols, 2003.

[42] V. C. Ravikumar, "EaseCAM: An Energy and Storage Efficient TCAM-Based Router Architecture for IP Lookup," vol. 54, ed: IEEE Computer Society, 2005, pp. 521-533.

[43] F. Zane, et al., "Coolcams: power-efficient TCAMs for forwarding engines," in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, 2003, pp. 42-52 vol.1.

[44] Ren Fengyuan, Lin Chuang, Liu Weidong, "Congestion Control in IP Networks," Chinese Journal of Computers, vol. 26, 2003.

[45] V. Jacobson, "Congestion avoidance and control," SIGCOMM Comput. Commun. Rev., vol. 25, pp. 157-187, 1995.

[46] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," RFC 2001, January 1997.

[47] S. Floyd and T. Henderson, "RFC2582: The NewReno Modification to TCP' s Fast Recovery Algorithm," RFC Editor United States, 1999.

[48] M. Mathis, et al., "RFC2018: TCP Selective Acknowledgement Options," RFC Editor United States, 1996.

[49] L. Brakmo, "TCP Vegas: End to end congestion avoidance on a global Internet," IEEE Journal on selected Areas in communications, vol. 13, p. 1465, 1995.

[50] S. Floyd, et al., "Adaptive RED: An algorithm for increasing the robustness of RED' s active queue management," Preprint, available at http://www.icir.org/floyd/papers.html, 2001.

[51] F. M. Anjum and L. Tassiulas, "Balanced-RED: An algorithm to achieve fairness in the Internet," in Proceedings of IEEE INFOCOM1999, New York, USA, 1999.

[52] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow-Control in Integrated Services Networks - the Single Node Case," Ieee Infocom 92 - the Conference on Computer Communications, Proceedings Vols 1-3, pp. 915-924 2515, 1992.

[53] H. Shimonishi and H. Suzuki, "Performance analysis of Weighted Round Robin cell scheduling and its improvement in ATM networks,"Ieice Transactions on Communications, vol. E81b, pp. 910-918, May 1998.

[54] A. Demers, et al., "Analysis and simulation of a fair queueing algorithm," ACM SIGCOMM Computer Communication Review, vol. 19, pp. 1-12, 1989.

Author Biographies

Luo Layong: Ph.D. candidate, Network Technology Research Center, Institute of Computing Technology, Chinese Academy of Sciences. luolayong@ict.ac.cn

Xie Yingke: Senior Engineer, Master' s Supervisor, Network Technology Research Center, Institute of Computing Technology, Chinese Academy of Sciences.

Xie Gaogang: Director of Network Technology Research Center, Professor, Ph.D. Supervisor, Institute of Computing Technology, Chinese Academy of Sciences.

Note: Figure translations are in progress. See original paper for figures.