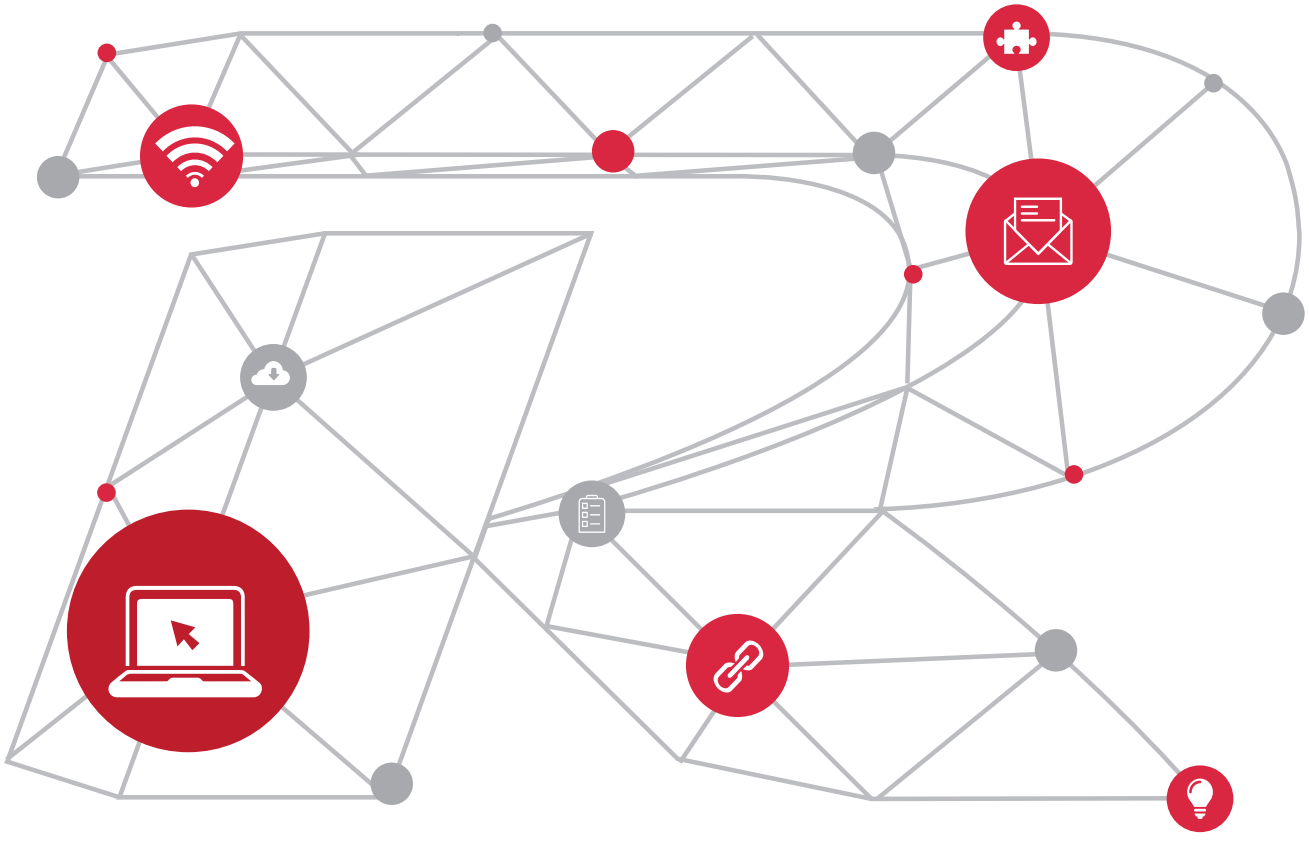
# Ruijie Express Forwarding

White Paper



# Contents

# Introduction

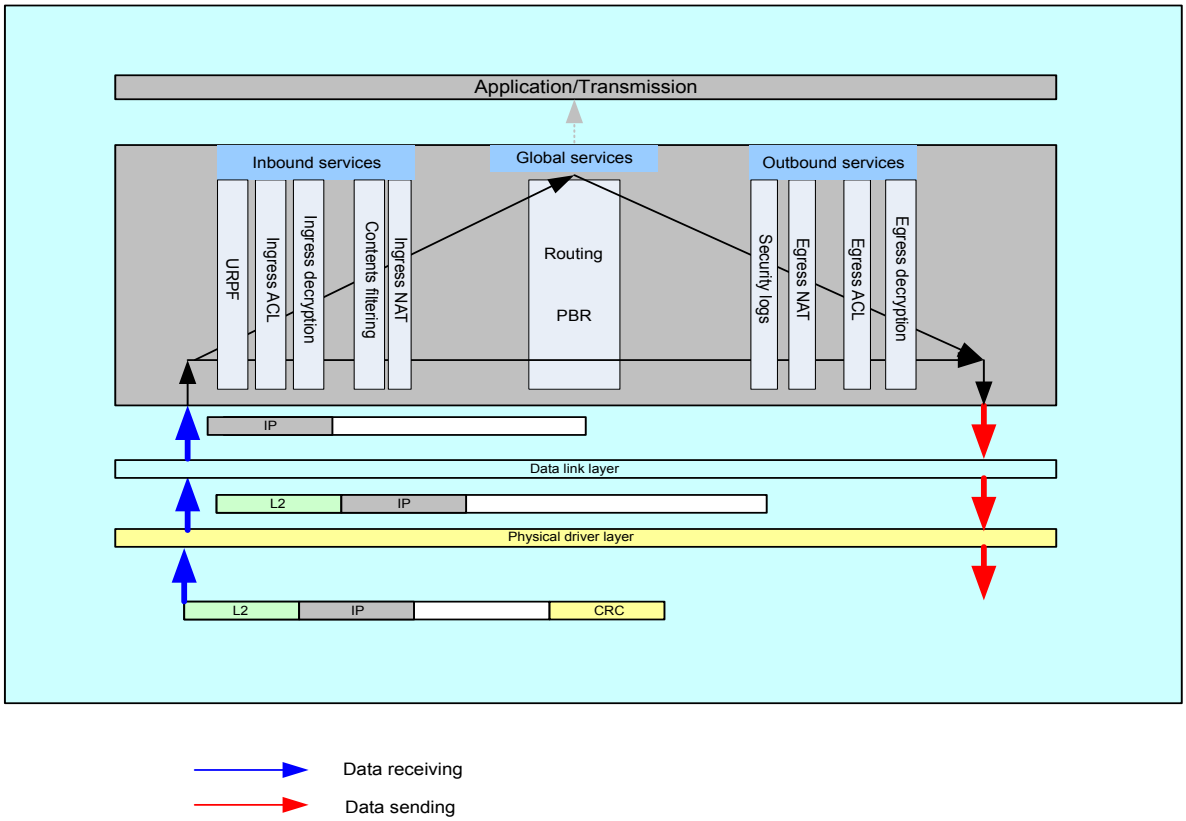
Ruijie Express Forwarding (REF) is a high-performance, multiservice, high-QoS, anti-attack, high-speed switching model that is based on Virtual CPU (VCPU) and class thread. It uses route management and flow table for fast search, and adopts cache management, mutual exclusion mechanism, and pipeline mechanism to achieve high CPU efficiency and high-speed service switching.

The most basic function of a router is to compute routes and forward packets. A traditional router uses a centralized CPU with a shared memory architecture to control the connection between the shared bus and multiple interface cards. The interface cards contain simple queues and communicate with the CPU and forward packets through the shared bus. As the Internet is developing rapidly and demands for more new services are growing, routers are required to provide higher capacity and switching performance, which means higher packet forwarding speed and service processing performance, to achieve faster service switching.

Being influenced by typical layered TCP/IP protocols, a traditional network device processes data packets on the data plane strictly based on layers: The physical driver layer receives a packet and checks the physical link; the packet enters the data link layer where its data link header is removed; it enters the network layer where IP processing is performed, such as resolution, checking, policy processing, security rule matching, routing, and delivering to the transport layer; the packet then enters the data link layer again where it is encapsulated with a link header; finally, it is sent in a queue on the physical driver layer. This forwarding mode is called process forwarding.

As a typical packet processing mode, process forwarding processes related network services. However, its design of processing packets via multiple processes and layered architecture significantly consume CPU resources. Therefore, the CPU cannot be used effectively and the data service forwarding performance is far from meeting the ever-increasing demands for higher service performance.

**Figure 1**




To eliminate the performance bottleneck of process forwarding, express forwarding is proposed. The first packet of the ones with the same IP address is sent to the processing module. Using the process forwarding technology, the module checks the routing table for the next-hop IP address and the destination network interface, and forwards the IP packet accordingly. During the course of forwarding, the information required to forward packets to the destination IP address is input into the express forwarding table of the express forwarding module. This information includes (but is not limited to) the destination IP address, next-hop IP address, L2 address for forwarding (MAC address of the outbound network interface for the Ethernet) and destination L2 address (destination MAC address obtained through Address Resolution Protocol (ARP) for the Ethernet). Later, when the router receives packets destined for the same IP address, it will not send them to the processing module. Instead, it performs express forwarding according to the express forwarding table within the packet reception break. The express forwarding table is a subset of the routing table and L2 address table.

The disadvantage of “one-stage routing, multi-stage switching” is that process switching is needed for each packet destined for an IP address to create express forwarding table entries cached on the router. A lot of IP packets destined for different IP addresses need to be routed for the first time when a network starts, resulting in congestion in the router. In the “one-stage routing, multi-stage switching” solution, express forwarding table entries maintain paths to hosts with different destination IP addresses (which constitute the host forwarding table) after packets are routed. The host forwarding table has a loose mapping relationship with the main routing table in the router. When the routing table or the ARP table changes, many entries in the express forwarding table will become invalid, and synchronizing the express forwarding table with the routing table will significantly affect the device performance. In an environment where high-speed dynamic routing is required (with frequent network topology changes, route changes, and route oscillation), such as the Internet, the caching mechanism of express forwarding cannot be flexible, route changes lead to invalid cache, and the overhead of rebuilding the cache (executing “process switching”) is high. As the Internet and its services are developing dramatically, web-based applications and interactive services increase, which produce large amounts of real-time data traffic for short but frequent sessions. The fast switching cache is constantly changing and the burden to rebuild the cache is getting heavy, which leads to lower router performance. The widely used security control exposes the weakness of the fast-switching cache mechanism, and the security control based on services cannot be provided.

To address the explosion of express forwarding table entries and to meet the multi-service demands, Cisco has launched Cisco Express Forwarding (CEF), to enhance routers' forwarding performance for high-speed service switching. How we can make use of a higher-performance multi-core CPU to meet the need for high-performance multi-service switching is one of the challenges that should be addressed on network devices today. Based on our solid understanding of Cisco's CEF and the practical situations of networks, Ruijie Networks has developed Ruijie Express Forwarding (REF) to maximize CPU performance for high-speed service switching.

# Concepts

* **Ruijie express forwarding (REF)**

It uses innovative technologies such as virtual cores and class thread to optimize data structure and make full use of CPU performance, so as to achieve express forwarding.

* **Virtual CPU (VCPU)**

It is a CPU virtualization method. In this document, the system core is virtualized into two cores: virtual data core and virtual system core. They run independently through a time slice scheduling mechanism.

* **Concurrent execution**

In a multi-service system, multiple tasks (processes or threads) are executed alternately.

* **Parallel execution**

In a multi-service system, multiple tasks (processes or threads) are executed simultaneously at any time. Parallel execution of multiple services can only occur in a multi-core CPU or multi-CPU system, and cannot occur in a single-core single-CPU system.

* **Express flow switching (X-FLOW)**

It organizes data in quintuples and works as the basic platform for switching services such as the access control list (ACL), policy-based routing (PBR), network address translation (NAT), and state firewall.

* **Forwarding information base (FIB)**

It uses the multi-branched tree to organize the information to be forwarded (mainly the routing prefix and routing behavior). It can locate the next-hop information rapidly and effectively by using more space to save time.

* **Adjacency table (ADJ)**

It is used to save the adjacency forwarding information (including the adjacent link information and adjacency behavior) to achieve fast replacement of packet headers for data forwarding.

* **Pipeline**

It is like working on an assembly line in a factory. All circuit units in a CPU with different functions form a pipeline for instruction execution. An instruction can be divided into many steps, which will be executed by these units, so instruction execution can be completed within a CPU clock cycle and the CPU speed is increased.

* **Cache**

It is a high-speed, small-capacity memory between the main memory and the CPU.

* **Mutual exclusion**

Only one operation can be performed at one time. This document mainly describes how the data plane, control plane, and management plane are mutually exclusive of one another.

* **Data plane**

Processing packets, including search in the FIB, X-FLOW decision, QoS policies, forwarding and discarding packets according to results, and processes.

* **Control plane**

It creates and maintains the data structure, including the FIB, X-FLOW, and QoS policies, used for forwarding packets.

* **Management plane**

It completes configuration management, statistics, and protocol packets processing, and protocol functions.

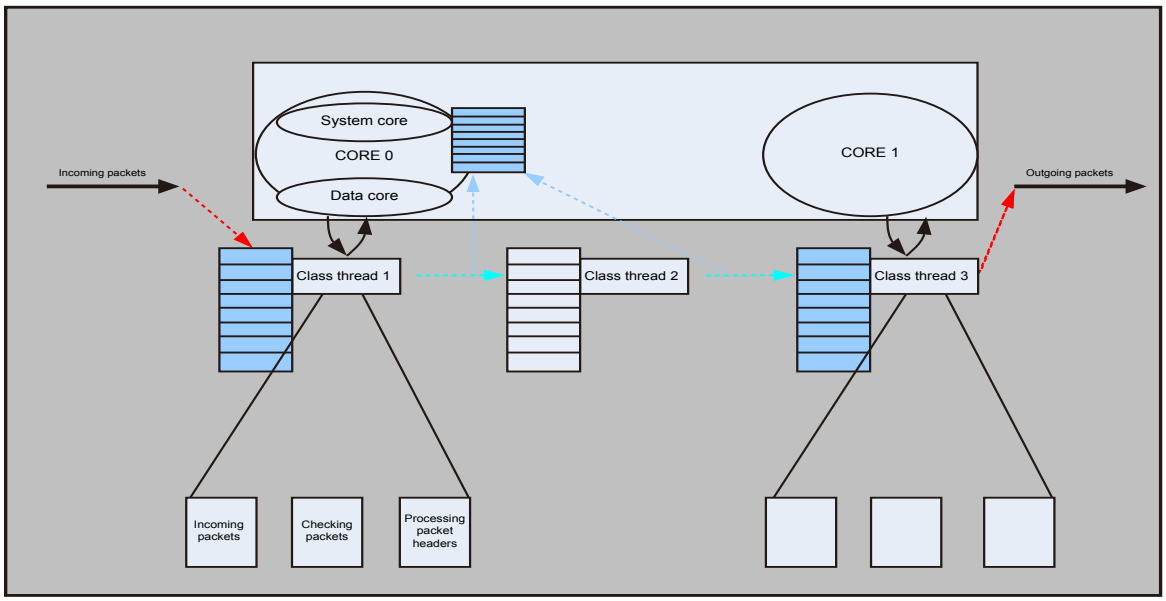
* **Class thread**

It is an allocation mode similar to a thread, the minimum unit for CPU resource allocation. A class thread can run on any VCPU. In this document, class threads refer to those on the express forwarding data plane.

# Technical Principle

## • Basics

**Figure 2**



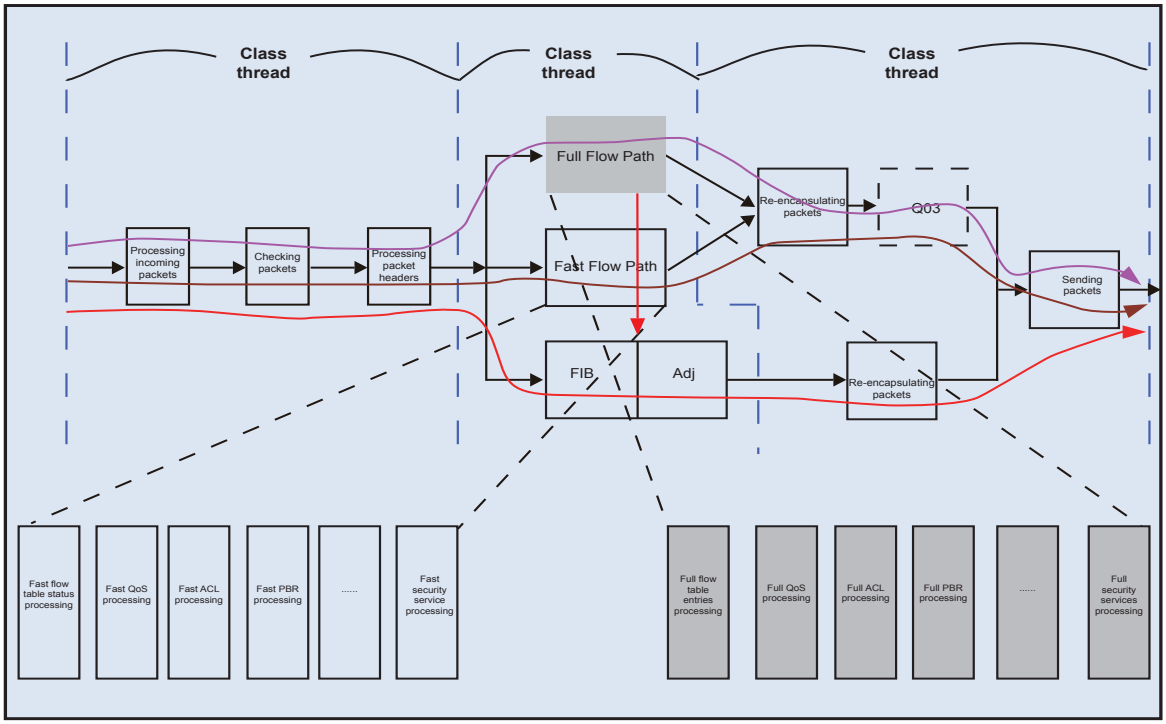
The foundation of REF is that one core in the multi-core CPU is virtualized into two cores: a virtual system core that is used for scheduling on the operating system (OS) layer, common TCP/IP protocol stack and its management, and work on the control plane; and a virtual data core, which is used for processes on the data plane, and is logically equivalent to the other core in terms of data processing. Data processing is divided into multiple class threads based on the CPU performance and service module processing time, and the packet link connects the class threads. Cores can process the class threads independently to achieve parallel processing to improve data processing efficiency.

Packets enter the system from a physical link and form a packet link through the driver. The system scheduling mechanism schedules the processing of class thread 1, which can run in any core as long as there is a free core. After class thread 1 is processed, the packet link is divided into three links: forwarding, discarding, and previous process. The forwarding link activates class thread 2 to make it ready, the previous process link activates previous processing, and the discarding link starts fast recovery based on the number of accumulated packets. From the above analysis we can see that in a multi-core system, class thread 1 can activate class thread 2; cores are equivalent; class thread 2 can run in Core1; packets enter the system continuously from a physical link; class thread 1 can run in Core0 if it is free; class thread 1 can parallel class thread 2 in different cores for pipeline operations to improve data processing performance.

The classification granularity of class threads is related to service processing and is adjusted according to the CPU performance, so as to balance the load among all class threads and avoid the situation where some threads are too busy while some are idle.

## • Data Flows

**Figure 3**



REF data flows are logically divided for three paths:

1. Express path. It is the simplest routing data packet switching, involving only route searching and no other operations during the processing of data flows.
2. Fast flow switching path. The flow table entries for this path have been created, and all service functions are achieved through fast processing. It is used for subsequent data processing after the traffic flow is established, and is the most common path for data flows in real network environments today.
3. Complete flow switching path. A complete flow switching path may include creation of flow table entries, service functions that are achieved through complete processing, and search of routing information. It emerges at the early stage when the traffic flow is established or when there are changes in configuration control (for example, ACL configuration change) and the routing information.

When a system is running, it chooses paths according to the flow status to avoid unbalanced processing time of different paths and unbalanced load among class threads, which affects forwarding efficiency. The flow path and class thread mechanism works with the mutual exclusion mechanism to make full use of the multi-core CPU performance. Different paths are independent of one another when they process services, enhancing the service switching performance.

The classification of class threads shown in the above figure is a schematic drawing, and it needs to be adjusted according to the CPU capability and service module processing performance in practice.

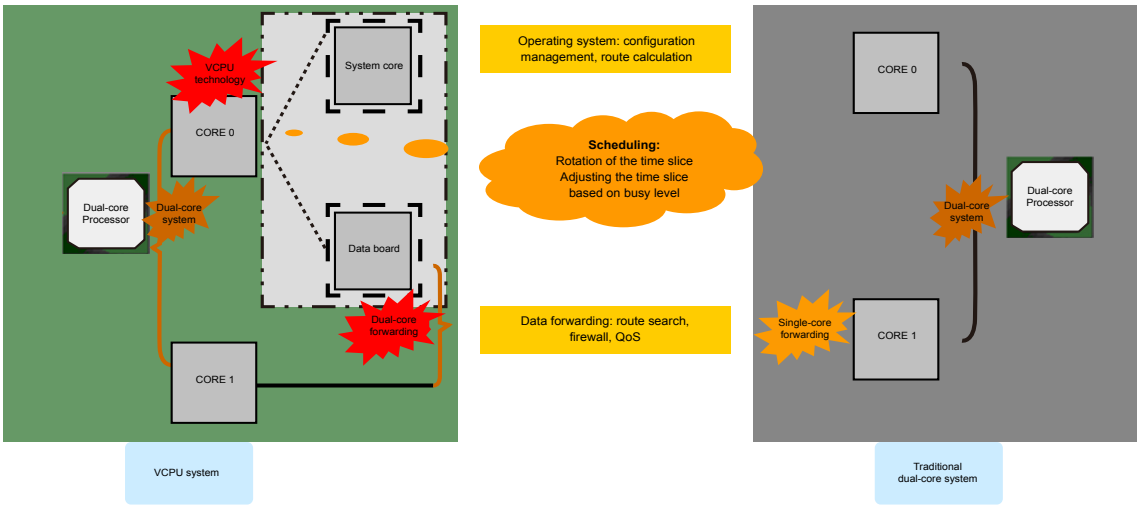
# Key Components

## • VCPU

The core where the OS runs is virtualized into two cores: virtual data core and virtual system core. The virtual data core runs the class threads on the express forwarding data plane, processes service modules and data packets on the express forwarding plane, and is equivalent to other data cores. The virtual system core executes processes in the OS, including processes on the control plane and management plane, and processes data. The virtual data core and virtual system core are actually running in one core and are virtualized using software. In a single-core system, one virtual core runs the express forwarding data plane and the other virtual core runs other parts. In a dual-core system, the virtual data core and the other core are used to achieve express data forwarding. The introduction of virtualization completely separates the data, control and management planes in a single-core system. A special scheduling mode independent of the OS effectively combines the data, control and management planes to facilitate high-speed data processing and ensure the priority of the management and control planes. In a multi-core system, data forwarding by cores is achieved besides the functions of a single-core system, which makes full use of the multi-core system performance to improve data processing capability.

Virtualization is scheduling and multiplexing by time slice. It is a solution that gives attention to both fairness and efficiency between two virtual cores. When both virtual cores are busy, make sure the load allocated to them fairly; when one virtual core is busy, the other gets more load. The system, according to the level of busyness of the virtual cores, defines the busy index, based on which it adjusts the ratio of their occupancy of the time slice. When the busy index of virtual cores is changed, the express forwarding core system adjusts dynamically the time division weight ratio of both virtual cores. Time division multiplexing is to adjust the time division weight ratio dynamically based on the busy index of both virtual cores. The principle of adjusting the weight is “maintaining dynamic balance based on feedback”, which means adjusting the use rate according to service features and proceeding to the next adjustment according to the feedback on the busyness of the adjusted virtual cores. To achieve effective time division multiplexing, selection of the time slice size is very important, and it should be adjusted based on the system processing capability for optimal effect. The virtual data core will make the best use of the system’s free time slice to ensure high-speed data processing. When the virtual system core is busy, the virtual data core will release resources to achieve management and control first.

**Figure 4**




## • Class Threads

In this document, a class thread refers to a “thread” that runs on the express forwarding data plane, is not supported by the OS, and cannot invoke any system functions in the OS. A class thread is global on the express forwarding data plane. When there is parallel execution of class threads on the express forwarding data plane in more than one core, one class thread will not be executed by two or more cores simultaneously. When a ready class thread is executed by a core through the scheduling routine, it is in the running status and removed from the queue of ready threads, so other cores cannot call it again.

A class thread is mainly used to process packet flows, and it has an incoming packet queue and an outgoing packet queue. When the system organizes the information of incoming and outgoing queues into a link, a pipeline for forwarding and processing packet flows is established. When a class thread is called to run, it only process the packets already in the incoming queue, and the packets that come in during the time when it is running will be processed the next time when it is called.

The system core is responsible for dynamic operation management of class threads on the whole data plane. A ready class thread is selected and put into a data core for execution. When the execution of a class thread at one time is completed, the data core will be called to execute another ready class thread. If multiple cores are free, they will execute multiple class threads simultaneously.

Status of class threads:

**Ready**: waiting for the CPU resource;

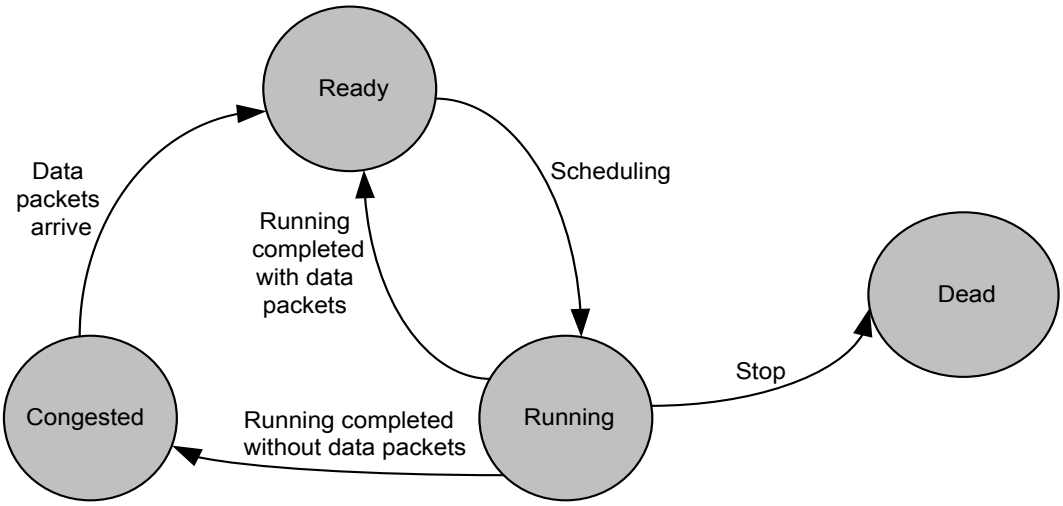
**Congested**: waiting for incoming packets;

**Running**: being executed by a core;

**Dead**: waiting for the resource to be released and itself to be deleted.

The following figure shows how the class thread status changes:

**Figure 5**




Management of class thread scheduling

Scheduling of class threads: based on the scheduling algorithm that combines priorities and the first come first served (FCFS) principle, ready class threads are divided into multiple queues. Class threads of different priorities are scheduled according to their priorities, and class threads of the same priority are scheduled according to the FCFS principle.

## • Mutual Exclusion Mechanism

For a memory to be accessed on the data plane, we should try to avoid using the mutual exclusion mechanism to protect the critical area. Processing packets one by one can significantly degrade the data forwarding performance. REF does is design to protect the critical data (involved in service modules) not through mutual exclusion but through other mechanisms.

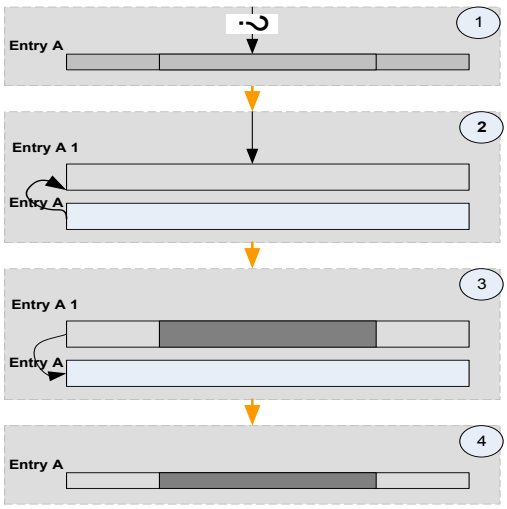
One of the methods used combines preprocessing with atomic operations.

The following figure shows the principle of the method:

1. The management and control planes need to perform operations on entry A, which is being used by the data plane.
2. To avoid mutual exclusion on the data plane, entry A is copied to entry A1, so operations by the management and control planes are transferred to entry A1. During this period, the data plane can still use entry A for processing, which is called preprocessing.
3. Atomic operations are performed to achieve interchangeable operations between entry A1 and entry A and to fulfill operations to entry A.
4. The data plane and the management and control planes use the updated entry A.

Revision and adjustment are needed to complete preprocessing and atomic operations for protection of the critical area without using mutual exclusion on the data plane according to actual situations in practice.

**Figure 6**



## • Cache Management

The principle of cache is based on the program access partiality. During a time period, the addresses generated by the program (including the instruction address and data address) are within a very small range of the memory address space. Frequent access to a partial range of memory addresses and hardly any access to addresses out of this range is called program access partiality.

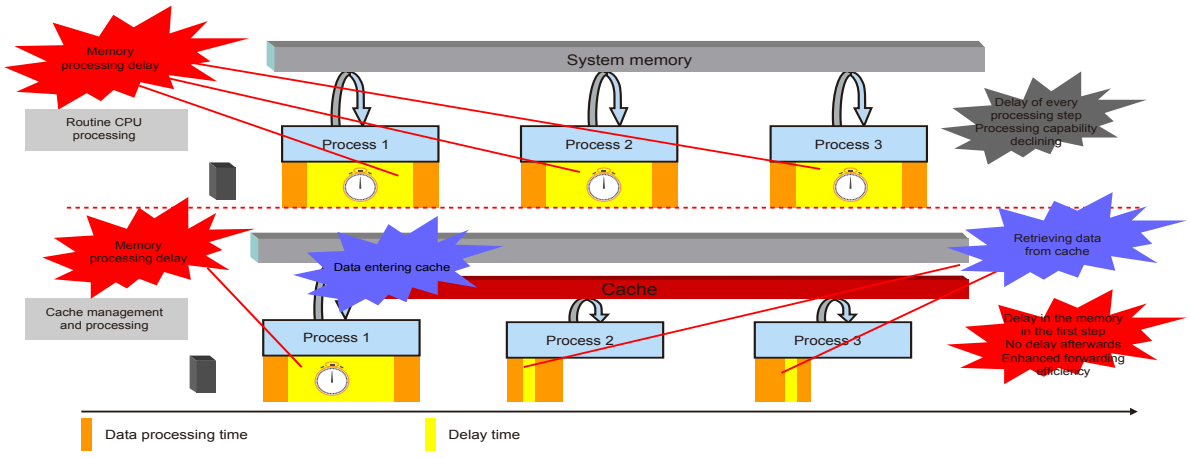
Based on the program access partiality, a high-speed, low-capacity memory is added between the main memory and the CPU, so that a segment of instructions or data near the address of the instruction being executed is transferred to this memory from the main memory for the CPU's use. This will reduce the times that a CPU retrieves data from the main memory, thus enhancing the operating speed. The system continuously reads a successive instruction set related to the current instruction set to the cache, which will be transferred at high speed by the CPU for high-speed matching. When the CPU requests data, it accesses the cache first. Due to the program access partiality, the cache can significantly speed up the CPU's processing of instructions and data.

When data packets are forwarded, the processing focuses on the header byte mostly. If the header byte is added to the cache, the overhead of accessing the memory during data processing can be reduced and data processing capability and forwarding speed can be enhanced. After analyzing the principles of data flows and the CPU cache mechanism, the header is included in the cache for packet processing based on service needs.

In terms of the design of flow table entries, they should be organized according the cache mechanism, so that the query of the flow table will be more efficient. Operations on the data plane should be included as many as possible in the same cache row to ensure that recent access to the CPU is performed in the cache.

In a modular data structure, the data plane is separated from the control plane to follow the cache management, so that the data on the data plane is as little as possible and the most commonly used data is concentrated in one cache row to improve the forwarding efficiency on the data plane.

**Figure 7**




## • Pipelining

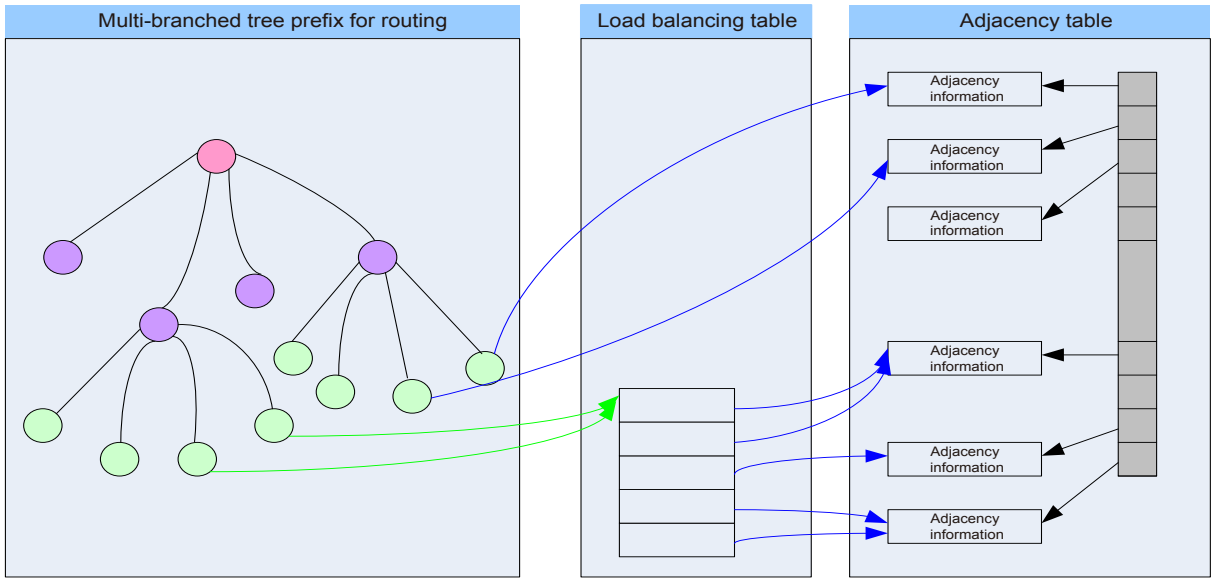
Pipelining is like working on an assembly line in a factory. All circuit units with different functions in a CPU form a pipeline for instruction execution. An instruction can be divided into many steps, which will be executed by these units, so instruction execution can be completed within a CPU clock cycle and the CPU speed is increased. If a CPU has multiple independent pipelines and uses them to execute multiple instructions, then it can fulfill the purposes of multiple instructions within a clock cycle.

There are two types of pipeline mechanisms in REF: one is multi-core pipeline, which is similar to the class thread mechanism; the other is an instruction set in a single-core program for pipeline optimization. They are implemented to enable the CPU to perform parallel processing. The class thread mechanism is elaborated previously, so it is not narrated here.

The following technologies are used to optimize the instruction set pipeline. Removing data correlation: the requesting RAM unit has correlated address data, so the CPU cannot perform parallel processing, which greatly reduces utilization of the CPU. To achieve pipelining of parallel processes, data correlation should be removed among instructions in an instruction set. Looping: the sensitivity of pipelines to branching statements decreases the operating efficiency. Looping can be adopted to achieve parallel processing according to features of CPU pipelining.

## • FIB + ADJ

**Figure 8**



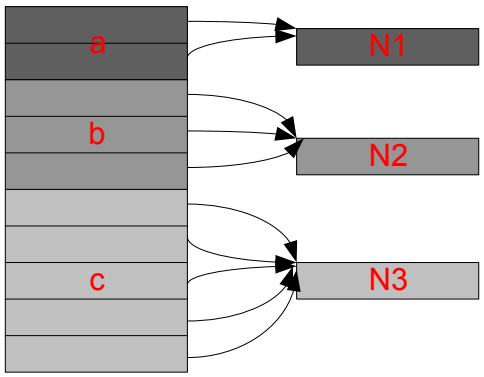

The FIB is a table in which the system searches for routes to the destination. An entry in the FIB is mapped to an entry in the IP routing table, which means the FIB is the mirroring of routing information in the IP routing table. As the FIB contains all necessary routing information, there is no need to maintain the routing cache. When the network topology is changed or the IP routing table is updated, the FIB is changed accordingly. To achieve fast search, the multi-branched tree is used to store the routing information and an advanced compression technology is used to save space, based on the “using more space to save time” theory and IP address features. Route searching based on IPv4 addresses take four times at most, which greatly increases the route searching speed and paves the way for high-speed switching.

When there are multiple routes to a destination IP address, with each route having a weight that reflects the cost, the routing protocol calculates the path to the destination address with the weight. The purpose of load balancing is to allocate traffic flows to multiple routes to optimize the use of resources. REF supports two types of load balancing: configuring load balancing based on the destination address and configuring load balancing based on the source/ destination address. The principle is that packets with a given (source or) destination IP address take the same route, even if there are multiple routes available; data flows with different (source or) destination address tend to take different routes. By balancing the load based on the (source or) destination address, you can make sure that data packets with a particular (source or) destination address arrive in sequence. When REF is enabled, load balancing based on the destination address is enabled by default.

The load balance table is introduced to support multipath routing to enhance searching efficiency and is implemented using the algorithm for approximately fair pre-storing in the switch chip. It exists when multipath routing is available. When the routing information is updated, the load balance table is updated and the switchover between multipath and single-path is updated. Algorithm principle: the next hops of route R are N1 (weight a), N2 (weight b), and N3 (weight c), and the load balance table is a bucket of depth of n. The load balance table is distributed in the ratio approximating a:b:c. Depth n of the load balance table determines its precision and needs to be adjusted according to actual situations to get to the target value.

REF uses the data link through which the adjacency table provides data packets to rewrite the necessary information. Each entry in the FIB points to the repeater section of a next hop in the adjacency table. If forwarding can be achieved between adjacent nodes, these nodes will be added to the adjacency table. Once the system finds the adjacency relationship, it writes this in the adjacency table, so the adjacency sequence is generated at any time. Every time an adjacent entry is generated, a link layer header is calculated for that adjacent node and it will be stored in the adjacency table. When the system is routing, it points to the next network segment and the corresponding adjacent route. The header will be encapsulated when the packets are ready for REF forwarding.

**Figure 9**




## • Express Flow Switching (X-FLOW)

X-FLOW enables extended processing based on quintuples to support the multi-service integration and improve performance. ACL, policy routing, NAT, firewall, and QoS are integrated usually. Using the concept “one-stage routing, multi-stage switching”, with the original flow aging and route interaction technologies, the impact of route changes to flows is relieved and therefore the IP network can be very flexible. The number of “short connections” is increasing in current services, and they work with the service module to achieve express flows and prevent the occurrence of new connection bottlenecks.

At least four categories of flows are defined according to protocols: TCP flows, UCP flows, ICMP flows, and RawIP flows.

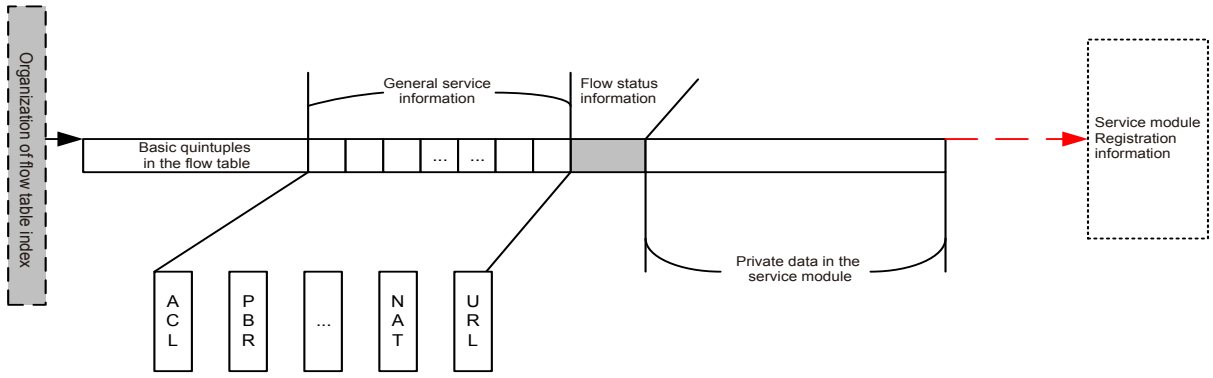
**TCP flows**: Source IP address, destination IP address, IP protocol, TCP source port, TCP destination port

**UDP flows**: Source IP address, destination IP address, IP protocol, UDP source port, UDP destination port

**ICMP flows**: Source IP address, destination IP address, IP protocol, ICMP ID, ICMP type and code

**RawIP flows**: Source IP address, destination IP address, IP protocol, user-defined, user-defined

**Figure 10**




Flow table entries provide a mechanism for express processing by recording information of flows. For example, in terms of ACLs, when a flow is established, it is unnecessary to compare each subsequent packet with the ACL, which significantly enhances the system performance, especially when there are a lot of rules. The flow table entries record the routing results, so the subsequent packets can be forwarded directly to achieve “one-stage routing, multi-stage switching.” Flow table entries track the data flow status and its identification of data flow is a filtering process based on the status, improving system and network security and anti-attack capability.

# Conclusion

REF integrates the processing of various aspects technically and adjusts the CPU features based on the service module processing performance to achieve express forwarding. Each feature in the REF technology is the basis of the REF performance, but the improvement of one feature cannot achieve express forwarding. All features need to work and balance with one another according to service needs to achieve overall express forwarding in a comprehensive way. To adapt to the increasing development of the IP network, the overall integration capability of REF is being enhanced.