Omada Switch CPP White Paper

Copyright © 2026 TP-Link Systems Inc. All rights reserved.

No part of this document may be reproduced, copied, or transmitted in any form or by any means without the prior written permission of the company.

Contents

1 Overview 3

1.1 Background 3

1.2 Technical Positioning 4

2 Key Technical Principles 4

2.1 Technical Overview 4

2.2 Principle Analysis 5

2.2.1 CPP Protection 5

2.2.2 Packet Identification and Filtering 6

2.2.3 Traffic Monitoring 7

2.2.4 Traffic Shaping 8

2.2.5 Dynamic Link Protection 10

3 Feature Implementation and Configuration Guide 10

3.1 CPP Configuration 10

3.2 Real-Time CPP Statistics 11

3.3 CPP Rate-Exceed Alarm 13

3.4 CPP Configuration Example 13

3.5 Test Data Comparison 14

4 Deployment Scenarios and Solutions 14

4.1 Daily Protection Scenarios 15

4.1.1 Enterprise Campus Access-Layer Protection 15

4.1.2 Aggregation/Core Gateway Protection 16

4.2 Attack Response Scenarios 17

4.2.1 Protocol Attack Identification and Mitigation 17

4.3 Service Assurance Scenarios 18

4.3.1 Firmware Updates and File Transfers 18

4.3.2 Transient Bursts During Topology Changes 19

5 Representative Models 20

6 Appendix 22

6.1 Glossary 22

1 Overview

CPP (CPU Protect Policy) is a hardware-based switch CPU protection technology. During network device operation, the CPU must process various protocol packets, management requests, and control-plane traffic. When faced with malicious attacks or sudden traffic bursts, CPU resources may be exhausted, leading to abnormal device responses or even service interruptions. CPP offloads traffic identification and rate limiting to the switching chip, enabling precise control of traffic destined for the CPU without consuming CPU resources, thereby ensuring stable device operation.

CPP's core capabilities:

- Hardware-based traffic processing makes the protection capability scale linearly with the ASIC forwarding performance without increasing CPU load as traffic grows.
- Three-level hierarchical rate-limiting model—CPU port-level control, CPU queue-level traffic separation and management, and protocol-level fine-grained rate limiting—provides a multi-layered defense system for the CPU.
- Dynamic link protection can temporarily open a higher bandwidth channel for specific protocols during designated service sessions and automatically restore the original rate-limiting policy after the session ends, balancing security and service availability.

1.1 Background

With the rapid development of intelligent networks, switches—core nodes of network infrastructure—are receiving increasing attention for their security and stability. In enterprise campuses, data centers, and service provider networks alike, the switch CPU, as the management core of the device, must continuously handle various control-plane messages, including routing protocol updates, spanning tree calculations, device management commands, and neighbor state maintenance.

However, as network scale expands and attack techniques evolve, switch CPUs face multiple threats:

- **Malicious attack traffic:** Attackers can send a large number of forged protocol requests or management packets to the device CPU, causing CPU overload. As a result, legitimate protocol packets cannot be processed, triggering cascading failures such as routing disruptions and management outages.
- **Fluctuations in normal service traffic:** In large networks, the density of legitimate protocol interactions can be high. During events such as topology changes or device restarts, control-plane traffic may spike instantaneously and exceed CPU processing capacity.
- **Lack of fine-grained control mechanisms:** Traditional software-based protection solutions (e.g., ACL- or firewall-based approaches) consume significant CPU resources

to execute matching logic. Under high volumes of attack traffic, they can further increase CPU load, creating a situation where the protection mechanism itself becomes the bottleneck.

To address these issues, protocol robustness alone is insufficient. Identification, rate limiting, and shaping must be performed in the hardware before packets reach the CPU, ensuring that the CPU always has sufficient resources to keep critical protocols running under all traffic conditions.

1.2 Technical Positioning

CPP is positioned in the field of switch control-plane security protection and serves as a foundation for ensuring CPU availability on network devices. Unlike traditional software-based protection methods, CPP implements packet matching and rate limiting in the switching chip's hardware system, providing the following characteristics:

- **Hardware-level processing efficiency:** Packet identification and rate limiting are completed in the hardware forwarding pipeline, without additional CPU interrupts or software processing overhead. This means CPP's protection effectiveness does not degrade as attack traffic increases.
- **Transparent deployment experience:** As a system-level feature operating at the device foundation layer, CPP executes protection actions automatically in the hardware once the administrator configures policies, without requiring additional adaptation at the service layer.
- **All-scenario policy coverage:** Through the three-level rate-limiting model and flexible mapping from applications to protocols, CPP supports a wide range of protection needs—from basic traffic limiting to fine-grained protocol-level control.

2 Key Technical Principles

2.1 Technical Overview

CPP on switches is designed based on a four-stage model (Classifying, Queueing, Scheduling, and Shaping), with enhancements to the Policing stage (protocol-level rate limiting). It forms a multi-level protection architecture that converges step by step through "protocol policing, queue shaping, and port shaping," with the following characteristics:

- **Hardware-based rate limiting:** Classification rules are programmed into the egress TCAM (EPCL) on the chip. Policing is enforced by the chip's egress policer, and Shaping

is enforced by the CPU port egress shaper. All processing is performed in the hardware.

- **Predefined protection policies:** After booting, the default policy is loaded automatically. Planned identification rules and rate limits are provided for 61 protocol types, ensuring baseline protection even without configuration.
- **Flexibility and adjustability:** Up to 12 user policy sets can be saved per scenario and switched quickly. Each protocol type can be configured independently to apply policing, drop, or collect statistics only (without policing).
- **Prioritized protection for critical traffic:** Rules are ordered by priority, ensuring that internal communications and critical protocols are identified first. Stacking inter-CPU communication uses dedicated no-limit entries to avoid unintended Policing.
- **No impact on operations:** Link protection temporarily increases bandwidth for the corresponding protocols during firmware upgrades and file transfers and automatically restores the original policy afterward. Identification rules are updated accordingly after the management ports are changed.
- **Observability:** CPP provides per-protocol pass/drop counters, rate, and packet loss ratio, as well as receive statistics for CPU hardware queues, and supports periodic packet-loss log alarms.

2.2 Principle Analysis

2.2.1 CPP Protection

CPP Protection is implemented in the hardware and does not consume CPU resources. It achieves strong scalability with packet handling (e.g., interception) capability unlimited by the CPU. CPP Protection mainly relies on two traffic control mechanisms: Single-Rate Three-Color Marker (SrTCM, RFC 2697) and Shaping. The former is used for protocol-level Policing, while the latter is used for CPU port and CPU queue rate limiting. Specifically:

- **Protocol-level policing:** After protocol matching, the matched packets are bound to the corresponding policer to enforce precise rate limiting.
- **Queue-based rate limiting:** Shaping is applied based on CPU hardware queues.
- **CPU port rate limiting:** Shaping is applied on the CPU port to perform unified traffic management for all packets punted to the CPU.

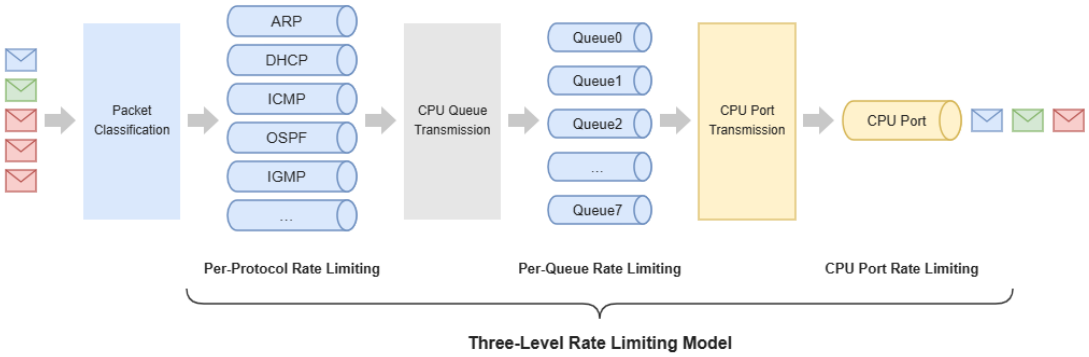


Figure 2.1 Three-Level Rate Limiting Model

CPU port shaping enforces a global rate ceiling on all traffic punted to the CPU at the control-plane ingress. Packets exceeding the threshold enter a hardware shaping buffer and are released smoothly at the target rate. It absorbs bursts through "buffering + delay," preventing peaks from saturating the CPU. CPU queue shaping maps control-plane traffic into CPU hardware queues by priority and configures shaping thresholds and bandwidth shares independently per queue. This ensures preferential protection for critical protocols (e.g., routing control-plane traffic) while constraining lower-value management traffic, preserving core control-plane processing capacity even under congestion. Protocol-level, fine-grained policing/dropping further classifies traffic by attributes such as MAC, EtherType, IP protocol number, and L4 ports. Independent token-bucket policers are bound to matched protocol classes, and suspicious or unnecessary packets can be denied directly, reducing invalid punts and shrinking the attack surface at the source.

2.2.2 Packet Identification and Filtering

The foundation of CPP's protocol-level precise rate limiting lies in the hardware's fast identification and matching of packets. The packet matching mechanism classifies packets based on protocol-specific characteristics (e.g., MAC address, IP address, EtherType, IP protocol number, and port number), and uses the corresponding hardware lookup mechanisms to filter packets that match these characteristics. Different actions are then applied to achieve traffic control for the specified packets.

CPP combines characteristic fields into match rules. As packets pass through, the hardware automatically extracts the relevant fields for comparison. When a rule is matched, the packet is processed according to the predefined action (e.g., rate limiting or dropping). The CPP packet matching and filtering process can be divided into the following four steps:

1) **Feature extraction:** The hardware extracts key fields from the packet, including source/destination MAC address, EtherType, IP protocol number, source/destination IP address, L4 port number, and more.
2) **Rule matching:** The extracted features are compared against the configured CPP

match rules. High-speed matching is performed via hardware table lookup. The matching process is completed in regular packet forwarding without introducing additional latency.

3) **Action enforcement:** After a match, the predefined action is applied. In rate-limiting mode, traffic within the threshold is forwarded normally, and excess traffic is dropped by hardware. In drop mode, packets matching the characteristics are dropped directly.
4) **CPU-bound traffic filtering:** CPP protects only packets sent to the CPU (e.g., routing protocol packets and management request packets). Normal forwarded data traffic is outside the CPU protection scope. The hardware performs a secondary confirmation by identifying the packet destination as the CPU interface, ensuring that rate limiting policies do not affect user data forwarding.

2.2.3 Traffic Monitoring

By configuring the parameters of the Single-Rate Three-Color Marker (SrTCM, a single-rate dual-token-bucket mechanism), CPP binds the corresponding rate limiter (meter) after matching the specified packets to perform traffic control. The hardware parameters involved include:

- **Committed Information Rate (CIR):** The token generation rate for the Committed Token Bucket (C bucket); i.e., the average rate at which packets are allowed to pass.
- **Committed Burst Size (CBS):** The capacity of the C bucket; i.e., the committed burst that can pass instantaneously.
- **Excess Burst Size (EBS):** The capacity of the Excess Token Bucket (E bucket); i.e., the allowed additional burst.

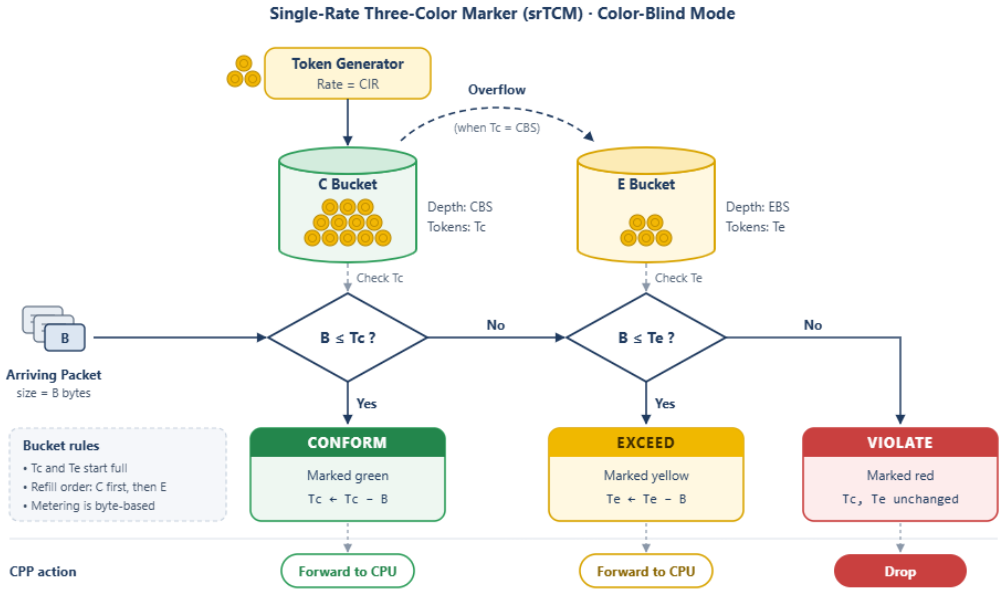


Figure 2.2: Single-Rate Three-Color Marker

When a packet arrives, the hardware determines the packet color according to its size and the current number of tokens in the buckets:

- **Green:** The packet size does not exceed the tokens in the C bucket. The packet is forwarded normally and consumes committed tokens.
- **Yellow:** The packet size exceeds the committed tokens but does not exceed the excess tokens. The packet is allowed to pass but is not guaranteed, and it consumes excess tokens.
- **Red:** The packet size exceeds the tokens in both buckets. Packets that exceed the rate limit threshold are dropped directly.

With this mechanism, CPP achieves precise traffic control that "allows short-term bursts but prevents long-term overload": short bursts within the CBS can pass normally, while sustained high-volume traffic exhausts tokens and causes packets to be dropped. On this basis, CPP collects statistics using two types of hardware counters:

- The CNC counter counts packets and total bytes for a specified protocol/class on the CPU port. Combined with the sampling interval, it can be used to calculate the receive rate.
- The Billing counter performs color-based statistics for SrTCM results. Packets marked red are dropped later by modules such as the egress policer, so the Billing counter can be used to derive metrics such as packet drops and packet loss ratio.

These statistics support real-time traffic monitoring. They also help determine whether packet loss is caused by rate limiting and whether control-plane traffic control parameters are reasonable, enabling faster troubleshooting.

2.2.4 Traffic Shaping

CPU port shaping and CPU queue shaping are implemented by shapers. The key difference compared with the SrTCM is buffering: packets that exceed the configured rate are not dropped immediately. Instead, they are buffered, and packets are dropped only when the buffer is full.

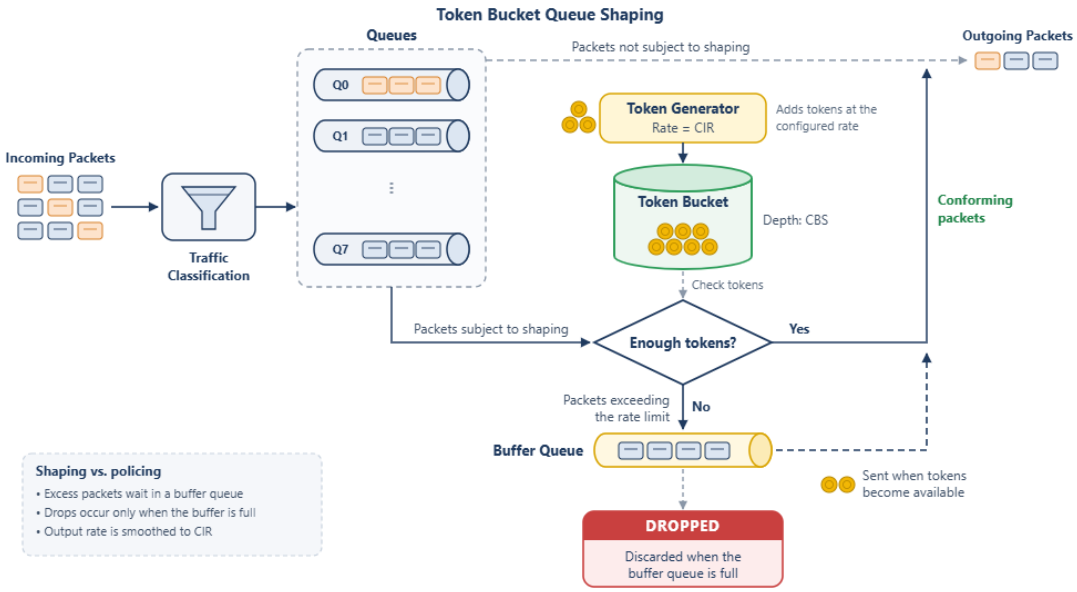


Figure 2.3: CPU Port Traffic Shaping

Traffic shaping buffers and queues packets that exceed the configured rate and transmits them gradually at a smoothed rate. It is suitable for overall traffic control on the CPU port and CPU queues. The workflow is as follows:

1) **Packet enqueuing:** After a packet reaches the shaper, it enters the corresponding buffer queue (CPU-port level or queue level) based on the configuration.
2) **Token check:** The hardware adds tokens to the token bucket at the configured rate. If enough tokens are available, the packet is sent immediately.
3) **Buffer queuing:** If tokens are insufficient, the packet waits in the hardware buffer queue. The system periodically dequeues packets from the buffer for transmission.
4) **Drop protection:** New incoming packets are dropped only when the buffer queue is full.

With shaping, when instantaneous bursts exceed the configured rate limit, packets are not dropped immediately. Instead, they are buffered and punted to the CPU at a steady rate. This is especially important in scenarios where legitimate bursts must be tolerated (e.g., routing updates during topology changes).

Table 2.1: Performance Comparison Between Traffic Monitoring and Traffic Shaping

Feature	Traffic Monitoring	Traffic Shaping
Exceeding Traffic Handling	Dropped immediately	Buffered
Application Level	Protocol policing	CPU port/queue
Typical Scenario	Precise blocking	Service bursts
Bandwidth Utilization	Bursts may be dropped	Bursts can be buffered

2.2.5 Dynamic Link Protection

In certain scenarios, the network device needs to temporarily provide higher CPU bandwidth to specific protocols (e.g., during firmware upgrades, batch configuration deployment, or large-scale file synchronization). If CPP remains constrained by its normal rate-limit thresholds in such cases, service efficiency may be reduced.

When a service module requiring temporary high bandwidth (e.g., an upgrade/configuration management module) initiates a request, CPP provides a temporary higher-rate channel for the corresponding protocol traffic on top of the current policy. After the service is complete, the initiating module notifies CPP to restore the original rate-limit state, and CPP reloads the original rate-limit parameters from the active policy. The bandwidth channel for Dynamic Link Protection has a higher priority than normal rate-limiting policies. This means:

- Eligible packets are served by the higher-bandwidth channel first.
- Other protocol packets outside the protection scope remain constrained by the original policy.
- After the session ends, the original rate-limiting policy is restored, and no permanent changes are made.

3 Feature Implementation and Configuration Guide

3.1 CPP Configuration

CPP is ready to use out of the box. After the device boots, CPP initialization loads the default-policy configuration by default. In the default-policy, different protocols have different rate-limit values. You can run the CLI command *show cpp default-policy* to view the default rate limits for various protocol packets. You can also create a custom policy to configure and apply rate limits for protocols, CPU queues, and the CPU port based on your network requirements.

CPP currently supports rate limiting for 61 protocol types. CIR can be set from 1 to 100,000,000 kbps. CBS can be set from 1 to 100,000,000 bytes.

rate-limit protocol [8021x|arp-bc|arp-uc|bfd|bgp|bgpv6|dns|dhcp-client|dhcp-server|dhcpv6-client|dhcpv6-server|dldp|efm|erps|ftp|tftp|gvrp|http|https|isis|lacp|lbd|ldp|lldp|mdns|mld-gmq|mld-gmrpt|mld-gmdone|mld-mlr|mpls-uc|mpls-bc|nd-rs|nd-ra|nd-ns|nd-na|nd-redirect|netbios|ospf|ospfv3|pim|pimv6|pppoe|ntp|ptp|pvst|radius|rip|ripng|stp|snmp|ssh|tacplus|telnet|vrrp|vrrp-v6|icmp|icmpv6|igmp|tcp-other|udp-other|other] cir [<1-100000000> cbs <1-100000000>| deny]

CPP also supports rate limiting for CPU queues 0–7. CIR can be set from 100 to

100,000,000 kbps. CBS can be set from 4 to 1023, in units of 4 KB. Queue 7 has the highest priority, and queue 0 has the lowest priority.

rate-limit queue-rate <0-7> cir <100-100000000> cbs <4-1023>

* Due to chip design limitations, some models (SG3452X/SG3452XP/SG2xxx) do not support the queue-related configurations above.

CPP supports rate limiting for the CPU port. CIR can be set from 100 to 100,000,000 kbps. CBS can be set from 4 to 1023, in units of 4 KB. If the CPU port rate limit is set too low, it may affect normal communication between stacked systems or MLAG systems.

rate-limit cpu-rate cir <100-100000000> cbs <4-1023>

*Due to chip design limitations, some models (SG3452X/SG3452XP/SG2xxx) do not support the CPU-port-related configurations above.

3.2 Real-Time CPP Statistics

CPP provides real-time statistics for different types of packets processed by the CPU, including:

- Cumulative counts of forwarded/dropped packets after the switch receives each type of packet
- Cumulative forwarded/dropped bytes
- Real-time rate and drop ratio

These statistics help users quickly understand the traffic conditions of different packet types in the network (at the current moment or over a period of time) and adjust CPP settings accordingly. In addition, CPP supports:

- Queue-based traffic statistics, helping users understand how packet types are scheduled by priority
- CPU-port-based traffic statistics, helping users understand the real-time number of packets/bytes delivered to the CPU and the corresponding rates

```
SG6654X(config)#show cpp statistics all
Packet-type     Pass(packets)  Drop(packets)  Pass(bytes)   Drop(bytes)   Rate(kbps)  Drop(%)
-----------     -------------  -------------  -----------   -----------   ----------  -------
8021x           0              0              0             0             0.00        0%
arp-bc          27175          140172890      3913200       20184896160   255.97      99%
arp-uc          446            23             64224         3312          0.00        4%
bfd             0              0              0             0             0.00        0%
bgp             0              0              0             0             0.00        0%
bgpv6           0              0              0             0             0.00        0%
dns             0              0              0             0             0.00        0%
dhcp-client     13             0              4745          0             0.58        0%
dhcp-server     0              0              0             0             0.00        0%
dhcpv6-client   0              0              0             0             0.00        0%
dhcpv6-server   0              0              0             0             0.00        0%
dldp            0              0              0             0             0.00        0%
efm             0              0              0             0             0.00        0%
erps            0              0              0             0             0.00        0%
ftp             0              0              0             0             0.00        0%
tftp            0              0              0             0             0.00        0%
gvrp            0              0              0             0             0.00        0%
http            0              0              0             0             0.00        0%
https           0              0              0             0             0.00        0%
isis            0              0              0             0             0.00        0%
lacp            0              0              0             0             0.00        0%
lbd             0              0              0             0             0.00        0%
ldp             0              0              0             0             0.00        0%
lldp            6              0              1698          0             0.00        0%
mdns            0              0              0             0             0.00        0%
mld-gmq         0              0              0             0             0.00        0%
mld-gmrpt       0              0              0             0             0.00        0%
mld-gmdone      0              0              0             0             0.00        0%
mld-mlr         0              0              0             0             0.00        0%
mpls-uc         0              0              0             0             0.00        0%
mpls-bc         0              0              0             0             0.00        0%
nd-rs           17             0              1530          0             0.14        0%
nd-ra           0              0              0             0             0.00        0%
nd-ns           0              0              0             0             0.00        0%
nd-na           0              0              0             0             0.00        0%
nd-redirect     0              0              0             0             0.00        0%
netbios         0              0              0             0             0.00        0%
ospf            0              0              0             0             0.00        0%
```

```
ospfv3          0              0              0             0             0.00        0%
pim             0              0              0             0             0.00        0%
pimv6           0              0              0             0             0.00        0%
pppoe           0              0              0             0             0.00        0%
ntp             0              0              0             0             0.00        0%
ptp             0              0              0             0             0.00        0%
rpvst           0              0              0             0             0.00        0%
radius          0              0              0             0             0.00        0%
rip             0              0              0             0             0.00        0%
ripng           0              0              0             0             0.00        0%
stp             39             0              3120          0             0.26        0%
snmp            0              0              0             0             0.00        0%
ssh             0              0              0             0             0.00        0%
tacplus         0              0              0             0             0.00        0%
telnet          0              0              0             0             0.00        0%
vrrp            0              0              0             0             0.00        0%
vrrpv6          0              0              0             0             0.00        0%
icmp            0              0              0             0             0.00        0%
icmpv6          0              0              0             0             0.00        0%
igmp            0              0              0             0             0.00        0%
tcp-other       0              0              0             0             0.00        0%
udp-other       0              0              0             0             0.00        0%
other           0              0              0             0             0.00        0%
```

```
SG6654X(config)#show cpp statistics cpu-rate
 cpu/queue |        rcvPackets         rcvBytes         Rate(kbps)
-----------+       -----------       ---------       ------------
 cpu       |        5219               1238797          1982.08
 queue0    |        0                  0                0.00
 queue1    |        0                  0                0.00
 queue2    |        0                  0                0.00
 queue3    |        0                  0                0.00
 queue4    |        1115               159445           255.11
 queue5    |        0                  0                0.00
 queue6    |        4104               1079352          1726.96
 queue7    |        0                  0                0.00
```

```
SG6654X(config)#show cpp statistics protocol arp-bc
Packet-type    Pass(packets)  Drop(packets)  Pass(bytes)   Drop(bytes)   Rate(kbps)  Drop(%)
-----------    -------------  -------------  -----------   -----------   ----------  -------
arp-bc         42730          258395558      6153120       37208960352   255.97      99%
```

3.3 CPP Rate-Exceed Alarm

When the traffic of a certain protocol type destined for the CPU bursts, exceeds the configured rate limit, and triggers packet drops, CPP prints a log message to the console port 10 minutes later to notify the user, enabling timely network anomaly detection.

Example:

2006-01-01 09:00:49,[CPP]/NOTICE(5)/The rate of critical packets [arp-bc] to CPU exceeded the configured rate limit, resulting in discards.

3.4 CPP Configuration Example

1. Create a custom CPP policy *test1*, configure rate limiting for the arp-bc protocol, and apply the policy.
2. Generate broadcast ARP request traffic and send it to a device port at a rate of 1 Gbps.
3. Run *show cpp statistics protocol arp-bc* to view the real-time limited rate and the number of forwarded/dropped packets.

```
SG6654X(config)#cpp-policy test1

SG6654X(config-cpp-policy)# rate-limit protocol arp-bc cir 512 cbs 64000

SG6654X(config-cpp-policy)#exit

SG6654X(config)#cpp-policy-apply policy test1
```

```
SG6654X(config)#show cpp statistics protocol arp-bc
Packet-type    Pass(packets)  Drop(packets)  Pass(bytes)   Drop(bytes)   Rate(kbps)  Drop(%)
-----------    -------------  -------------  -----------   -----------   ----------  -------
arp-bc         72502          401882149      10440288      57871029456   512.18      99%
```

The feature of creating and applying a custom policy is currently supported only in Standalone mode. It is not supported in Controller mode at this time. If you need to use this feature, you can deploy the configuration using a CLI template.

3.5 Test Data Comparison

Protocol Type	Test Guide	CPP Supported	CPU Utilization	Manageable via SSH	STP Calculation in Ring Topology
ARP	DUT: Default configuration. Tester: Create an ARP Request Streamblock with destination MAC = all Fs and destination IP in the same subnet as the interface IP, and continuously send it to any DUT port at 1,000,000 kbps.	No	4%->88%	Yes	Normal
		Yes	4%->14%	Yes	Normal
STP	DUT: Default configuration (RSTP enabled globally by default; ports have spanning tree enabled by default). Tester: Create a BPDU Streamblock with destination MAC = 33:33:00:00:00:01 and continuously send it to any DUT port at 1,000,000 kbps.	No	4%->85%	Yes	Abnormal (network storm may occur)
		Yes	4%->10%	Yes	Normal
DHCP	DUT: Default configuration (VLANIF1 obtains an IP address via DHCP). Tester: Create a DHCP Discover Streamblock with destination MAC = all Fs and continuously send it to any DUT port at 1,000,000 kbps.	No	4%->81%	Yes	Normal
		Yes	4%->12%	Yes	Normal

4 Deployment Scenarios and Solutions

As attacks continue to evolve, switch CPUs face increasingly severe security threats. These threats range from protocol weaknesses (e.g., Spanning Tree attacks and routing protocol attacks), to malicious traffic surges (e.g., ARP flooding and worms), and even legitimate burst

traffic. Under these combined pressures, the switch CPU must still keep critical protocols running reliably.

CPP is a core control-plane protection mechanism for switches. By classifying packets destined to the CPU, applying priority scheduling, and enforcing bandwidth rate limiting, CPP ensures that the CPU always has enough resources to handle critical tasks, which helps deliver a stable network experience.

CPP deployment scenarios fall into three categories: daily protection, attack response, and service assurance. The following sections describe each category.

4.1 Daily Protection Scenarios

4.1.1 Enterprise Campus Access-Layer Protection

Scenario Characteristics:

Access-layer switches connect directly to end users and serve as the first line of defense. Common threats include:

- An endpoint infected with a worm (e.g., Slammer) generating large volumes of ARP requests or unknown unicast scans
- A rogue DHCP server sending spoofed DHCP Offer packets
- Broadcast storms causing large volumes of packets to be sent to the CPU

Risk Analysis:

Using ARP attacks as an example: tests show that without CPP enabled, sending broadcast ARP Request packets at 1,000,000 kbps can increase CPU utilization from a normal 4% to 80%+. Although you might still be able to manage the switch over SSH and STP computation might still work, the CPU is close to saturation. Any additional protocol processing can cause abnormal device behavior.

Recommended CPP Protection Policy:

Setting	Recommended Value	Description
ARP Protocol Rate Limit	256–400 kbps	Based on test results, CPP can suppress ARP attack traffic sent to the CPU from 1,800 kbps down to 256 kbps.
DHCP Protocol Rate Limit	Rate limit as needed (512 kbps)	Prevent rogue DHCP server traffic from overwhelming the CPU.
STP (BPDU)	128 kbps	Prevent Spanning Tree attacks from causing

Setting	Recommended Value	Description
Protocol Rate Limit		topology instability.
CPU Port Shaping	Configure based on device specs	Caps total CPU-bound traffic globally.

Note: Chip performance varies by model. The values in this document are recommendations and must be adjusted based on the actual model and production requirements. The same applies to recommended values in later sections.

Expected Results:

With CPP enabled, under the same ARP attack conditions, CPU utilization increases only from 4% to 14%. Switch management remains available, and STP computation remains normal. By enforcing rate limiting in the switching chip, CPP effectively suppresses attack traffic and preserves CPU capacity.

4.1.2 Aggregation/Core Gateway Protection

Scenario Characteristics:

Aggregation or core switches often act as user gateways and must handle:

- Large-scale ARP table maintenance
- Dynamic routing protocols (e.g., OSPF, BGP, RIP, etc.)
- Multicast protocols (e.g., PIM and IGMP)
- Device management traffic (e.g., SNMP, SSH, and Telnet)

Risk Analysis:

During topology changes or routing protocol attacks, large volumes of protocol packets can be sent to the CPU at the same time. Without priority scheduling, critical packets (e.g., OSPF Hello and BGP Keepalive) might time out due to CPU overload. This can flap neighbor relationships and destabilize routing across the network.

Recommended CPP Protection Policy:

Protocol Type	Rate-Limiting Policy	Description
STP (BPDU), VRRP, GVRP, etc.	Higher rate limits (512–2,048 kbps)	Highest-priority protection to keep the base topology stable.

Protocol Type	Rate-Limiting Policy	Description
OSPF, BGP, RIP, PIM, etc.	Higher rate limits (512–2,048 kbps)	Routing stability directly affects end-to-end reachability.
ARP, DHCP, IGMP, etc.	Medium rate limits (512–3,072 kbps)	Basic access services for users.
SNMP, SSH, Telnet, ICMP, etc.	Lower rate limits (256–1,024 kbps)	Management traffic; maintain manageability.
Unknown-IPMC, others	Strict rate limits or drop (64–256 kbps)	Non-critical traffic; reduce the attack surface at the source.

Expected Results:

By combining queue priority scheduling with protocol-level policing, critical protocol packets are processed first even when CPU load is high. This helps keep topology and routing protocols stable and prevents flapping and route churn.

4.2 Attack Response Scenarios

4.2.1 Protocol Attack Identification and Mitigation

Scenario Characteristics:

A malicious actor or malware-infected host sends large volumes of spoofed packets by emulating specific protocols. Common protocol attacks include:

- **Spanning Tree attacks:** Spoofed BPDU packets trigger frequent topology recalculation and network-wide instability
- **Routing protocol attacks:** Spoofed route updates or large volumes of protocol packets cause route flapping
- **ARP flood attacks:** Spoofed ARP requests exhaust CPU resources
- **TTL=1 attacks:** Large volumes of TTL=1 IP packets force the switch CPU to process them

Risk Analysis:

Using BPDU attacks as an example: tests show that without CPP enabled, sending BPDU packets at 1,000,000 kbps can increase CPU utilization from 4% to 85%. More critically, STP computation can become abnormal, potentially leading to a network storm and a full outage. This happens because spoofed BPDU packets consume CPU resources and prevent the CPU from correctly processing legitimate STP traffic.

Expected Results:

With CPP enabled under the same conditions, CPU utilization rises only to 10%, STP computation remains normal, and the topology stays stable.

CPP Attack Response Workflow:

Step	Action	Description
1. Identify the attack	Run *show cpp statistics** to review packet counts and drop counts by packet type. *Supported show commands may vary by model. Refer to the actual device.	Identify the protocol type with the traffic spike.
2. Enforce rate limits	CPP hardware automatically drops packets that exceed the configured limit.	Configure appropriate rate limits in advance.
3. Log and alert	CPP generates an alert log after sustained packet drops for 10 minutes.	Notify administrators to investigate the anomaly.
4. Trace and remediate	Use the statistics to locate the attacking source port or IP.	Block the source with ACLs or port security features.
5. Verify recovery	After the attack stops, confirm CPU utilization returns to normal.	Adjust the rate-limit thresholds as needed.

4.3 Service Assurance Scenarios

4.3.1 Firmware Updates and File Transfers

Scenario Characteristics:

Network administrators may need to update switch firmware or push configurations in batches, typically transferring files over HTTP, HTTPS, or TFTP. If standard CPP policies apply strict rate limits to these protocols, updates can take too long and reduce operational efficiency.

Dynamic Link Protection Solution:

Phase	Action	Description
Before update	The update module requests a temporary rate increase from CPP.	Opens a higher-bandwidth path for specific protocols based on the existing policy.

Phase	Action	Description
During update	CPP temporarily increases the HTTP/HTTPS/TFTP rate-limit thresholds.	Transfers update files faster and shortens the maintenance window.
After update	The update module notifies CPP to restore the original rate limits.	Automatically restores settings without leaving permanent policy changes.

Benefits:

This dynamic link protection mechanism balances security and availability: strict rate limits during normal operation for safety, and temporary bandwidth increases during specific tasks for efficiency. It enables a flexible "lock down by default, open when needed, close when done" policy.

4.3.2 Transient Bursts During Topology Changes

Scenario Characteristics:

During topology changes (e.g., link failures, device restarts, or STP convergence), legitimate protocol traffic can spike briefly:

- Dense LSA update exchanges within an OSPF area
- STP Topology Change Notification (TCN) flooding
- Large-scale ARP table refreshes

CPP Mechanisms:

Mechanism	Description
Traffic Shaping	CPU port and queue shapers include buffering. Excess packets are queued in the buffer instead of being dropped immediately.
CBS (Committed Burst Size)	CBS in the token bucket allows short bursts to pass and tolerates legitimate transient peaks.
Queue Priority Scheduling	Higher-priority protocol packets are processed first.

Differences from Traffic Policing:

Mechanism	Traffic Policing	Traffic Shaping
When Traffic	Drop immediately	Buffered and smooth traffic

Exceeds the Limit		
Best For	Precisely blocking attack traffic	Tolerating legitimate burst traffic
Scope	Protocol level	CPU port level and queue level

5 Representative Models

Starting in 2026, non-EOL models within the Omada and Omada Pro series—including those in the Campus, Aggregation, Access Max, Access Pro, Access Plus, and Access Switch categories—will gradually gain support for the CPP feature via software updates.

The Campus Switch models that already have CPP-supported firmware released include SG6428X, SG6428XHP, SG6654X, SG6654XHP, S6500-48MPP6Y, S6500-24MPP6Y, S6500-48GP6XF, S6500-48G6XF, S6500-24GP6XF, and S6500-24G6XF. Some L2+ switch models also support CPP after being adapted to Controller versions 6.1 and later. For details, refer to the Release Notes for the corresponding models on the official website.

For the product series mentioned above, the following models are planned to support CPP in the future:

Model	Protocols Supported for Rate Limiting	CPU Queue Rate Limiting/CPU Port Rate Limiting Supported
S7500-32C S7500-48Y8C S7500-24Y4C S7500-48XF4C S7500-26XF6Y S6500-48MPP6Y S6500-48M6Y S6500-24MPP4Y S6500-24M4Y S6500-48GP6XF S6500-48G6XF S6500-24GP4XF S6500-24G4XF SX6632YF SG6654XHP SG6654X SG6428XHP	8021x, arp-bc, arp-uc, dns, dhcp-client, dhcp-server, dhcpv6-client, dhcpv6-server, dldp, erps, ftp, tftp, gvrp, http, https, lacp, lbd, lldp, mdns, mld-gmq, mld-gmrpt, mld-gmdone, mld-mlr, nd-rs, nd-ra, nd-ns, nd-na, nd-redirect (nd-rr), netbios, ntp, pppoe, rpvst, radius, rip, stp, snmp, ssh, tacplus, telnet, icmp, icmpv6, igmp, tcp-other, udp-other, bfd, bgp, bgpv6, efm, isis, ldp, mpls-uc, mpls-bc, ospf, ospfv3, pim, pimv6, ptp, ripng, vrrp, vrrpv6	Yes

SG6428X		
SG3452XP SG3452X SG5452XMPP SG5452X SG5428XMPP SG5428X SG5428XF SG3452XMPP S5500-48GP4XF S5500-48G4XF SX3032F SX3832 SX3832MPP SX3016F S5500-16XF SX3008F SG3428XF SX3206HPP SG3428XMPP SG3428XMP SG3428X S5500-4XHPP2XF S5500-8XF S5500-24F4XF S5500-24GP4XF	8021x, arp-bc, arp-uc, dns, dhcp-client, dhcp-server, dhcpv6-client, dhcpv6-server, dldp, erps, ftp, tftp, gvrp, http, https, lacp, lbd, lldp, mdns, mld-gmq, mld-gmrpt, mld-gmdone, mld-mlr, nd-rs, nd-ra, nd-ns, nd-na, nd-redirect (nd-rr), netbios, ntp, pppoe, rpvst, radius, rip, stp, snmp, ssh, tacplus, telnet, icmp, icmpv6, igmp, tcp-other, udp-other	No
SG3428XPP-M2 SG3428X-M2 SG3218XP-M2 SG3210XHP-M2 SG3210X-M2 SG2210XMP-M2 SG3452P SG3452 SG3428MP SG3428 SG3210 SG2452LP SG2428P SG2428LP SG2218P SG2218 SG2016P SG2210MP		

SG2210P SG2008P SG2008 SG2206MP SG2005P-PD S5500-24MPP4XF S5500-8MHP2XF S5500-48GP4F S5500-24GP4F S4500-16GP S4500-16G2F S4500-8GHP2F S4500-8GP2F S4500-8GP S4500-8G		

6 Appendix

6.1 Glossary

Term	Full Name	Description
CPP	CPU Protect Policy	A hardware-based switch CPU protection technology that offloads traffic identification and rate limiting to the switching chip.
CPU	Central Processing Unit	The core unit in a switch that processes protocol packets, management requests, and control-plane communications.
TCAM	Ternary Content-Addressable Memory	A hardware lookup mechanism for high-speed rule matching on the chip. It's used for egress EPCL.
EPCL	Egress Policy Control List	Classification rules are installed in the chip's egress TCAM and used for protocol matching and rate limiting.
SrTCM	Single-Rate Three-Color Marker	A traffic-control mechanism based on a single-rate, two-bucket token bucket (RFC 2697), used for protocol-level rate limiting.
CIR	Committed Information Rate	The rate at which tokens are added to the C bucket. It defines the average rate at which packets are allowed to be

Term	Full Name	Description
		transmitted.
CBS	Committed Burst Size	The capacity of the C bucket. It defines the maximum committed burst that can pass instantaneously.
EBS	Excess Burst Size	The capacity of the E bucket. It defines the allowed excess burst.
C Bucket	Committed Token Bucket	The bucket that stores tokens corresponding to the committed rate.
E Bucket	Excess Token Bucket	The bucket that stores tokens corresponding to the excess burst.
Policing	—	The protocol-level rate limiting stage in the CPP four-stage model.
Classifying	—	Stage 1 of the CPP four-stage model, where packets are classified.
Queueing	—	Stage 2 of the CPP four-stage model, where packets are mapped to CPU hardware queues based on priority.
Scheduling	—	Stage 3 of the CPP four-stage model, where queues are scheduled.
Shaping	—	Stage 4 of the CPP four-stage model, where traffic is smoothed through buffering.
CNC Counter	—	Used to count total packets and bytes for specific protocols/classes on the CPU port.
Billing Counter	—	Tracks SrTCM color-marking results for each color, used to calculate packet loss and loss rate.