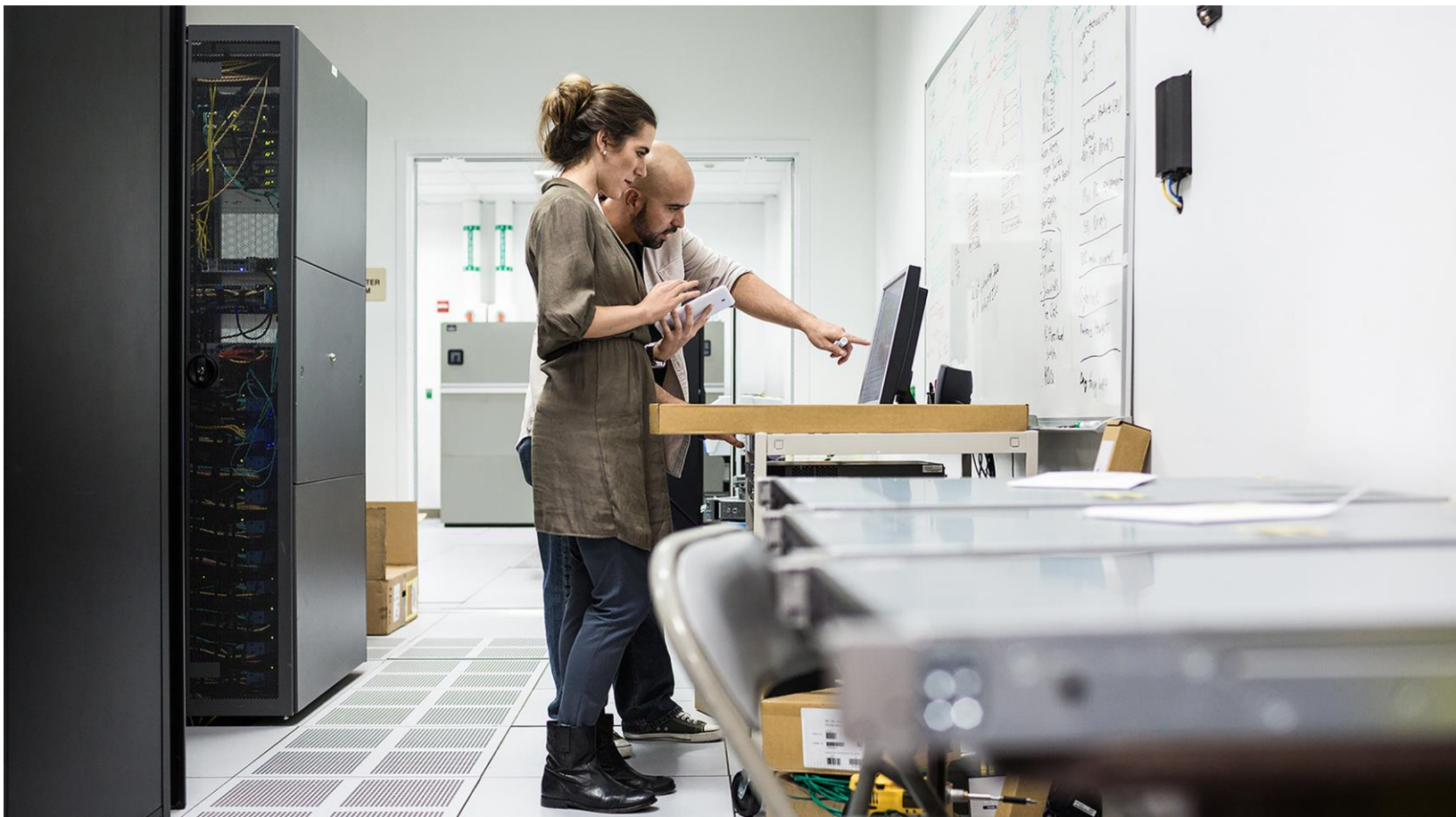


# Intelligent Buffer Management on Cisco Nexus 9000 Series Switches



---

# Contents

## **Introduction**

## **Prerequisites**

## **Hardware and Software Information**

## **Data Center Traffic Forwarding and Buffer Management**

- Data Center Flow Distribution and Forwarding Requirements
- Challenges for Switch Platforms with Simple Buffer Management
- Cisco Nexus 9000 Series Switches with Intelligent Buffer Management

## **Approximate Fair Dropping (AFD) with Elephant Trap**

- Elephant Trap (ETRAP)
- AFD Algorithm
- AFD Compared to RED and WRED
- AFD with Explicit Congestion Notification (ECN)
- Configuring AFD
  - Configuring ETRAP Parameters
  - Configuring Queuing Policy with AFD
  - Applying Queuing Policy with AFD to Interfaces
- Monitor AFD Counters

## **Dynamic Packet Prioritization (DPP)**

- DPP Mechanism
- Configuring DPP
  - Configuring DPP Parameters
  - Configuring Network-QoS Policy with DPP
  - Configuring and Applying Queuing Policy with DPP Mice Queue

## **Application Benefits of AFD and DPP**

- Cisco Intelligent Buffer Management with Elephant Flows
- Cisco Intelligent Buffer Management with Mice Flows

## **Conclusion**

## **Appendix A: TCP Congestion Control**

## **Appendix B: Sizing Buffers in Networks**

## **References**

---

## Introduction

Data center applications have been rapidly evolving in recent years. The new computing models, such as big data and real-time streaming processing, and application systems, such as in-memory computing and distributed machine learning, have changed workload and host communication patterns. Server-to-server traffic is dominant in a modern data center. Intensive concurrent communications can occur among tens to hundreds or even thousands of servers. Heavily fan-in traffic patterns and microbursts are often seen with distributed applications. All these new trends are demanding stringent throughput, latency, and scalability performance from data center networks. New data center switches need to be designed to meet these new requirements. One important element in the design of new switches is buffer architecture and management. Switches must be able to accommodate mixed intensive communication traffic flows with full throughput and low latency while also efficiently handling instantaneous bursts and microbursts to deliver excellent application performance.

This document discusses the new intelligent buffer management functions in Cisco® cloud-scale application-specific integrated circuits (ASICs) for Cisco Nexus® 9000 Series Switches. Cisco intelligent buffer management includes approximate fair dropping (AFD) with elephant trap (ETRAP), and dynamic packet prioritization (DPP) functions. It uses an algorithm-based architectural approach to address the buffer requirements in modern data centers. It offers a cost-effective and sustainable solution to support the ever-increasing network speed and data traffic load.

## Prerequisites

Readers of this document should be familiar with basic data center networking, network quality of service (QoS), network queuing and buffering, TCP/IP, and TCP congestion-control mechanisms. Appendixes A and B provide basic information about buffering in networks and TCP congestion control.

## Hardware and Software Information

Cisco intelligent buffer management functions are available on the Cisco Nexus 9000 Series Switches based on Cisco cloud-scale ASICs, including Nexus 9200, Nexus 9300-EX series fixed-form-factor switches, and the N9K-X9700-EX series line cards for the Nexus 9500 modular switches.

The software support for the Cisco intelligent buffer management functions is introduced in the NX-OS release 7.0(3)I5(1).

## Data Center Traffic Forwarding and Buffer Management

This section provides an overview of traffic-forwarding and buffer-management mechanisms in the data center.

### Data Center Flow Distribution and Forwarding Requirements

To fully understand the essence and benefits of Cisco intelligent buffer management, start by considering the characteristics and dynamics of traffic in modern data center networks.

Data center traffic is often characterized and measured in terms of flows. A flow corresponds to a sequence of packets from an application source host to a destination host. Each flow is uniquely identified by canonical five-tuple sequence: source IP address, destination IP address, source port, destination port, and protocol.

---

Extensive studies show that modern data center traffic is bimodal, consisting of short flows, often called mice flows, and long flows, often called elephant flows (see references 3, 4, and 5 at the end of this document). The exact distribution of flow lengths varies from data center to data center, but research shows that the distribution of flow lengths in data center networks is generally heavy tailed. About 80 to 90 percent of flows in data center networks are short and low-rate mice flows. The remaining 10 to 20 percent are elephant flows. However, only 10 percent of the data traffic load is contained in mice flows. In another words, a small number of elephant flows carry most (90 percent) of the data traffic. Mice flows add bursts to the overall traffic load. It is not uncommon for modern data center applications to create incast, also known as microburst, traffic patterns: essentially aggregations of a large number of mice flows. Elephant and mice flows perform different application jobs and so have different requirements for bandwidth, packet loss, and network latency.

Elephant flows correspond to application tasks that perform large data transfers: typically backup, virtual machine migration, and data structure synchronization across massively distributed systems. Because they transfer bulk data, they are aggressive in obtaining network bandwidth. The forwarding of elephant flows aims to achieve high throughput while keeping buffer occupancy low. When multiple elephant flows share the same congested link, the bandwidth allocation to each flow shall be controlled with ideally max-min fairness.

Mice flows consist of queries, responses, and control messages that are exchanged between the application hosts. These jobs are expected to complete within a minimal time, so mice flows are latency sensitive. In TCP applications, mice flows do not tolerate packet drops well. Because of the small number of packets in mice flows, most mice flows complete in the TCP slow-start phase, with a small TCP window size. There may not be enough packets in transit to trigger TCP fast retransmission for a lost packet. In this case, the sender will not be able to detect the packet loss until the TCP retransmission timer expires. Retransmission timeout can be detrimental to mice-flow performance and can severely degrade overall application performance. Therefore, it is imperative to avoid packet losses in mice flows. Appendix A provides more information about the TCP congestion-control mechanism and fast retransmission.

Given the distributed architecture of modern applications and the intensive communications among a large number of application hosts, inevitably a great number of elephant flows and mice flows will be mixed in the same queue on a network link. Their different forwarding requirements should be factored into the queue management scheme. Mice flows should be protected from packet losses and long queuing latency during link congestion. Elephant flows should be able to achieve high throughput with fair bandwidth allocation, but they should be prevented from causing buffer deficiency for mice flows by occupying too much of the buffer space.

These goals can be achieved on a switch by providing adequate buffer space for elephant flows to sustain full throughput on a congested link, yet restraining them from further queue occupancy by using TCP's own congestion-control mechanism. Appendix A describes the TCP congestion-control mechanism. Appendix B explains how to size buffers in a network for full throughput. Essentially, after the buffer contains enough packets to sustain 100 percent throughput on the congested link, buffering more packets does not offer additional bandwidth benefits. If senders of elephant flows continue to increase their transmit rate to send more packets into the network, the packets will end up waiting in the queue and consuming more buffer space. A better approach is to signal the senders to slow down by intentionally injecting a small number of packet drops in the flows. With TCP fast retransmission used for quick packet-loss detection and recovery, this small number of packet drops will have little impact on the completion time of the elephant flows. This approach will help control the buffer space for elephant flows so that the switch can reserve enough buffer headroom to absorb more mice flows, which will significantly improve the flow completion time of mice flows. As a combined result, the average flow completion time for all flows will be improved, leading to better overall application performance.

---

## **Challenges for Switch Platforms with Simple Buffer Management**

The approach to handling link congestion described in the preceding section requires switches to treat mice flows and elephant flows differently in a queue. However, traditional data center switches with simple buffer management are incapable of differentiating mice flows and elephant flows. All flows in a queue are subject to the same first-in-first-out (FIFO) scheduling, and the same queue-management scheme, such as random early detection (RED) or weighted random early detection (WRED). But FIFO queue scheduling does not prevent mice flows from experiencing long queuing latency during congestion or tail drops upon buffer overflow. RED and WRED are active queue-management mechanisms to alleviate congestion by using proactive packet drops to trigger TCP congestion control on the senders. However, RED and WRED randomly drop packets in a queue when the queue depth exceeds the congestion threshold, which can also affect mice flows, so it is not an ideal solution for protecting mice flows.

To avoid packet losses in mice flows, traditional switches need to buffer all packets at a congested link because they cannot selectively buffer mice flows. For this reason, some new switch platforms are built with deep buffers to help ensure ample buffer space. Aside from the significantly higher cost of a switch with greater buffer space, an immediate downside of this approach is the longer queuing latency caused by the extended queue depth. The latency will increase the flow completion time for all data flows, consequentially jeopardizing overall application performance. Additionally, even if the switch intends to buffer all packets during congestion, a small number of long-lived large elephant flows potentially can use up the entire available buffer space, leaving no buffer space for mice flows. TCP is a “greedy” protocol by design, with the tendency to fill up the links and buffers. Unless explicit congestion notification (ECN) is enabled, TCP uses packet drops as implicit congestion notifications to trigger its congestion-control mechanism on the sender to self-pace-down packet transmission. If no packet drops are detected, a TCP sender can continuously increase its transmit rate and keep sending packets to the network, which can eventually cause buffer overflow on the congested link. When this occurs, new mice flows arriving at the queue will be subject to tail drop. Therefore, despite the costly excessive buffer resources, the simple deep-buffer approach does not provide an ultimate solution to handle link congestion for both elephant flows and mice flows.

## **Cisco Nexus 9000 Series Switches with Intelligent Buffer Management**

Cisco Nexus 9000 Series Switches with Cisco cloud-scale ASICs are built with a moderate amount of on-chip buffer space to achieve 100 percent throughput on high-speed 10/25/40/50/100-Gbps links and with intelligent buffer management functions to efficiently serve mixed mice flows and elephant flows. The critical concept in Cisco's innovative intelligent buffer management is the capability to distinguish mice and elephant flows and apply different queue management schemes to them based on their network forwarding requirements in the event of link congestion. This capability allows both elephant flows and mice flows to achieve their best performance, which improves overall application performance.

The intelligent buffer management capabilities are built in to Cisco cloud-scale ASICs for hardware-accelerated performance. The main functions include approximate fair dropping (AFD) with elephant trap (ETRAP) and dynamic packet prioritization (DPP). AFD focuses on preserving buffer space to absorb mice flows, particularly microbursts, which are aggregated mice flows, by limiting the buffer use of aggressive elephant flows. It also aims to enforce bandwidth allocation fairness among elephant flows. DPP provides the capability of separating mice flows and elephant flows into two different queues so that buffer space can be allocated to them independently, and different queue scheduling can be applied to them. For example, mice flows can be mapped to a low-latency queue (LLQ), and elephant flows can be sent to a weighted fair queue. AFD and DPP can be deployed separately or jointly. They complement each other to deliver the best application performance. The following sections explain how AFD and DPP work and discuss their benefits for data center applications.

---

## Approximate Fair Dropping (AFD) with Elephant Trap

AFD is an active queue-management scheme whose fundamental goal is to provide fair bandwidth allocation among flows that share a common egress queue. The fairness has two aspects. First, AFD distinguishes long-lived elephant flows from short-lived mice flows and exempts mice flows from the dropping algorithm so that mice flows will get their fair share of bandwidth without being starved by bandwidth-hungry elephant flows. Second, AFD tracks elephant flows and subjects them to the AFD algorithm in the egress queue to grant them their fair share of bandwidth. The following sections describe the way that AFD functions.

### Elephant Trap (ETRAP)

AFD uses ETRAP to distinguish long-lived elephant flows from short-lived mice flows. A flow may be defined using multiple parameters, but typically the 5-tuple is used. ETRAP operates on the ingress side of a switch. It measures the byte counts of incoming flows and compares this against the ETRAP threshold that is defined in bytes. Flows with a byte count lower than the threshold are considered to be mice flows. After a flow crosses the threshold, it becomes an elephant flow and is moved to the elephant table for tracking. The ETRAP threshold is user configurable to allow customer traffic patterns to dictate what constitutes an elephant flow.

The elephant table stores and tracks elephant flows. Elephant flows in the table are measured to determine their data arrival rate and their activeness. The measured data rates are passed to the buffer management mechanism on egress queues, where the rates are used by the AFD algorithm to calculate the probability of drops for each flow. Elephant flows are aged out if they do not remain a certain level of activeness. A user-configured age-period timer and a bandwidth threshold are used to evaluate the liveness of an elephant flow. When its average bandwidth during the age-period time is lower than the configured bandwidth threshold, an elephant flow is considered to be inactive and will be timed-out and removed from the elephant flow table.

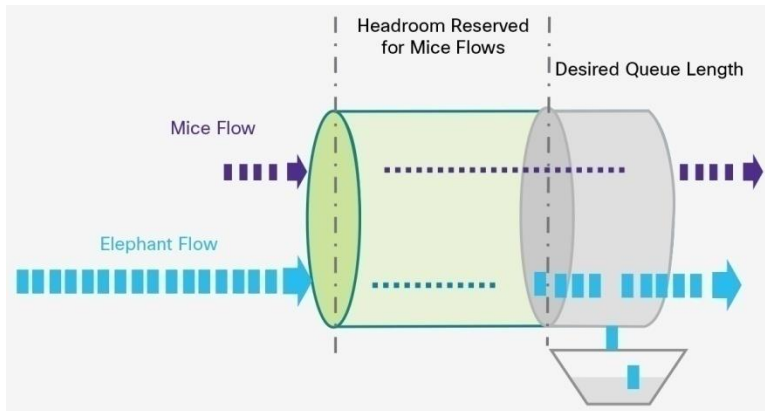
### AFD Algorithm

The central feature of the AFD algorithm is the intelligence to allocate bandwidth fairly among elephant flows based on their data rates. This feature has two main elements:

- Data rate measurement: ETRAP measures the arrival rate of each elephant flow on the ingress port and passes it to the buffer management mechanism on the egress port.
- Fair rate calculation: The AFD algorithm dynamically computes a per-flow fair rate for an egress queue using a feedback mechanism based on the occupancy of the egress port queue. When a packet of an elephant flow enters the egress queue, the AFD algorithm compares the measured arrival rate of the flow with the computed per-flow fair-share rate. If the arrival rate is less than the fair rate, the packet will be queued and eventually transmitted to the egress link. If the arrival rate exceeds the fair rate, the packet will be randomly dropped from that flow in proportion to the amount by which it exceeds the fair rate. So the drop probability is computed using the fair rate and the measured flow rate. The more a flow exceeds the fair rate, the higher the drop probability. As a result, all elephant flows achieve the fair rate.

Figure 1 depicts the overall effect of AFD. By submitting only elephant flows to the early-discard algorithm, AFD is able to prevent unwanted packet drops in mice flows and preserve enough buffer headroom to accommodate bursts particularly caused by a large number of simultaneous mice flows (incast traffic or microbursts). Among long-lived elephant flows, the AFD algorithm provides fair bandwidth allocation by applying fair early discards based on the data arrival rates.

**Figure 1.** AFD Algorithm



### **AFD Compared to RED and WRED**

AFD is a flow-aware early-discard mechanism that signals network congestion and engages the TCP congestion mechanism on the application hosts by dropping packets. Prior to AFD, weighted random early discard, or WRED, was the primary technology used for the same or similar purposes. WRED applies weighted random early-discard to class-based queues, but it doesn't have flow awareness within a class. All packets, including packet-loss-sensitive mice flows, are subject to the same drop probability. Hence, packets from mice flows are as likely to be dropped as packets from elephant flows. Although elephant flows can use drops as congestion signals to back off, drops can have a detrimental effect on mice flows. In addition, the same drop probability causes elephant flows with a higher rate (due to short round-trip time [RTT]) to obtain more bandwidth. Therefore, egress bandwidth is not evenly divided among elephant flows traversing the same congested link. As a result, the flow completion time for mice flows deteriorates, and elephant flows do not have fair access to the link bandwidth and buffer resources. In contrast, AFD takes into account the flow sizes and data arrival rates before making a drop decision. The dropping algorithm is designed to protect mice flows and to provide fairness among elephant flows during bandwidth contention.

### **AFD with Explicit Congestion Notification (ECN)**

By default, AFD applies early packet drops to elephant flows based on the dropping possibility calculated by the AFD algorithm. The packet drops serve as implicit congestion notifications to the source hosts. If it is desired to avoid packet drops, AFD and explicit congestion notification, or ECN, can be enabled together. AFD can work jointly with ECN to send ECN explicit congestion notifications without dropping packets. On the basis of the marking possibility calculated by the AFD algorithm, the ECN function on the switch can mark the ECN Congestion Experienced (CE) bit in packets. After that, the packets will continue to be transmitted toward the destination host instead of being dropped. The CE bit will be carried all the way to the receiving host. The receiving host will echo this congestion indication back to the sender using the ECN-echo (ECE) flag in the TCP header. When the source host receives the packet from the receiving host with the ECE flag set, it will reduce its transmission window in the same way as for a packet drop. In this way, AFD can provide the intended functions, including fair bandwidth allocation and network congestion mitigation, without dropping packets by using ECN.

### **Configuring AFD**

AFD configuration involves the following steps:

1. Define the ETRAP parameters.
2. Activate AFD in a queuing policy map by defining the desired queue depth.
3. Apply the queuing policy to the egress interfaces.

## Configuring ETRAP Parameters

Three ETRAP parameters need to be defined:

- **byte-count:** This parameter defines the threshold to identify elephant flows. When a flow starts transmitting more bytes than the ETRAP **byte-count** value, it becomes an elephant flow.
- **age-period:** This parameter sets the aging timer for inactive elephant flows.
- **bandwidth-threshold:** Together with **age-period**, **bandwidth-threshold** determines whether an elephant flow needs to be timed out. An elephant flow is considered idle and will be removed from the ETRAP flow table if it receives fewer bytes than **bandwidth-threshold** for the duration of **age-period**.

The default settings of these ETRAP parameters in the Cisco NX-OS Software Release 7.0(3)I5(1) are shown here. With this default configuration, flows receiving more than 1048555 bytes (approximately 1 MB) will be identified as elephant flows. An elephant flow will time out from the ETRAP flow table if it has less than 500 bytes data over a period of 500 microseconds.

```
hardware qos etrap age-period 500 usec
hardware qos etrap byte-count 1048555
hardware qos etrap bandwidth-threshold 500 bytes
```

**Note:** The default settings may vary in future NX-OS releases for fine-tuning. Please refer to the proper release notes and configuration guide for current information. Users also should adjust the parameters according to their data center traffic patterns.

## Configuring Queuing Policy with AFD

AFD itself is configured in queuing policies and applied to the egress class-based queues. The only parameter in a queuing policy map that needs to be configured for AFD is the desired queue depth for a given class-based queue. This parameter controls when AFD starts to apply algorithm-based drop or ECN marking to elephant flows within this class. AFD can be defined in any class-based queues. In the following example, AFD is applied to the default queue with a desired queue depth of 150 KB.

```
n9k# sh policy-map type queuing afd_8q-out

policy-map type queuing afd_8q-out
class type queuing c-out-8q-q7
priority level 1
class type queuing c-out-8q-q6
bandwidth remaining percent 0
class type queuing c-out-8q-q5
bandwidth remaining percent 0
class type queuing c-out-8q-q4
bandwidth remaining percent 0
class type queuing c-out-8q-q3
bandwidth remaining percent 0
class type queuing c-out-8q-q2
bandwidth remaining percent 0
class type queuing c-out-8q-q1
bandwidth remaining percent 0
class type queuing c-out-8q-q-default
  afd queue-desired 150 kbytes
  bandwidth remaining percent 100
n9k#
```

In this example, AFD is enabled in the default queue with a desired queue length of 150 KB.



The desired queue depth should be set differently for different link speeds of the egress port because it needs to be sufficient to achieve 100 percent throughput. It also should be a balance of the buffer headroom that needs to be reserved for mice flows, the number of packet retransmissions, and queue latency. Table 1 shows the recommended values for some typical link speeds, but users can choose different values in their particular data center environments.

**Table 1.** Recommended Desired Queue Depth for Typical Link Speeds

Port Speed	Value of Desired Queue Depth
10 Gbps	150 KB
40 Gbps	600 KB
100 Gbps	1500 KB

### Applying Queuing Policy with AFD to Interfaces

A queuing policy with AFD can be applied to the system QoS to make it the global queuing policy for all ports on the switch, or such a policy can be selectively applied to individual switch ports on which queuing policy with AFD is needed. Examples for both options are shown here:

```
• Applying AFD Queuing Policy to System QoS:  
n9k(config)# system qos  
n9k(config-sys-qos)# service-policy type queuing output afd_8q-out  
  
• Applying AFD Queuing Policy to Interfaces:  
n9k(config)# int e1/1  
n9k(config-if)# service-policy type queuing output afd_8q-out
```

Apply queuing policy with AFD to system QoS.

Apply queuing policy with AFD to interface e1/1.

Also, users can apply one policy with AFD at the system level and a different policy on an interface. The two can co-exist, with interface policy having higher priority. The policy at the system level will apply to all the interfaces, and the policy on an interface will overwrite the system-level policy for that particular interface. This flexibility allows easier and better control of AFD policies on a switch when an organization wants to apply different policies on some of the interfaces than those on most of the other interfaces.

## Monitor AFD Counters

After a queuing policy is applied to an interface, the **show queuing interface** command output shows the AFD counters in addition to the queue information.

```
n9k# show queuing interface e1/1

slot 1
=====

Egress Queuing for Ethernet1/1 [Interface]
-----
QoS-Group#  Bandwidth%  PrioLevel          Shape          QLimit
              Min          Max          Units
-----
      7         -         1         -         -         -         9 (D)
      6         0         -         -         -         -         9 (D)
      5         0         -         -         -         -         9 (D)
      4         0         -         -         -         -         9 (D)
      3         0         -         -         -         -         9 (D)
      2         0         -         -         -         -         9 (D)
      1         0         -         -         -         -         9 (D)
      0        100         -         -         -         -         9 (D)
+-----+
|                QOS GROUP 0                |
+-----+
|                | Unicast          | Multicast          |
+-----+
|                Tx Pkts |          2523844 |          0 |
|                Tx Byts |        645096506 |          0 |
| WRED/AFD Drop Pkts |          149786 |          0 |
| WRED/AFD Drop Byts |        38345468 |          0 |
|          Q Depth Byts |          247504 |          0 |
| WD & Tail Drop Pkts |          149786 |          0 |
+-----+
```

## Dynamic Packet Prioritization (DPP)

Dynamic packet prioritization, or DPP, provides the capability to manage mice flows and elephant flows in the same traffic class separately for queuing. It effectively isolates them in two separate queues despite the fact that they belong to the same traffic class. This separation is not possible with the traditional simple queue management technologies because they lack flow awareness within a class.

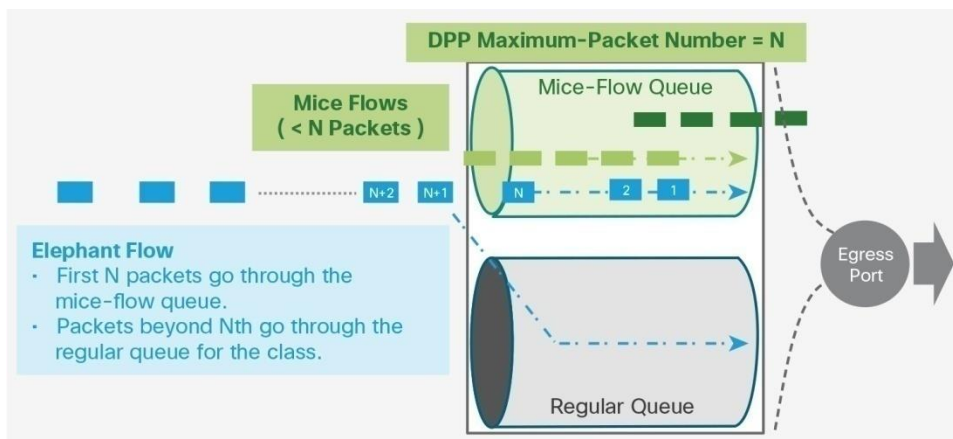
## DPP Mechanism

DPP uses a simple mechanism to achieve its function. This mechanism tracks flows to distinguish mice flows from elephant flows on the ingress port and then applies separate queuing policies to mice flows and elephant flows on the egress port.

On the ingress port, flows are identified by their five-tuples, and their initial packet counters are monitored. On the basis of the flow size, packets in a flow are then classified into the mice QoS group or the original QoS group for the traffic class. Within the DPP mechanism, a maximum-packet counter is used to determine the classification. If the DPP maximum-packet counter is defined as  $N$ , the first  $N$  packets in every flow are classified into the mice QoS group, and the  $N+1$ th packet and onward are classified into the original QoS group of the traffic class to which the flow belongs. Essentially, a new flow always starts as a mice flow until it sends more packets than the value of the DPP maximum-packet counter.

After DPP differentiates mice and elephant flows into separate QoS groups on the ingress port, different queuing controls can be applied to them on the egress port to meet the different forwarding requirements for mice flows and for elephant flows. Figure 2 depicts the DPP mechanism.

**Figure 2.** DPP Mechanism with Maximum-Packet Counter



The  $N$  value for the maximum-packet counter should be carefully chosen based on the traffic pattern in a data center network so that it can effectively separate mice flows and elephant flows. That is, it should be set to capture the mice flows or bursts in the mice-flow queue while leaving most of data load of the elephant flows in the regular queue.

The mice flow queue should be a priority queue while the regular queue for elephant flows is a weighted queue with bandwidth allocated for the needs of these flows. This is because of two reasons. First of all, with most of data center applications, mice flows are very sensitive to packet loss and network latency, whereas elephant flows have higher demands for bandwidth. Making the mice flow queue a priority queue can guarantee prioritized low-latency forwarding to mice flows. Secondly, applying priority queuing to the mice queue also prevents packets re-ordering in elephant flows because the initial packets in elephant flows are sent to the mice queue as well. The flexibility of separately serving mice flows and elephant flows in the same class makes it possible to simultaneously provide low-latency benefits for short-lived mice flows and sustain full link bandwidth utilization for elephant flows, which can significantly improve overall application performance.

To reserve buffer headroom for mice flows or to provide bandwidth fairness among elephant flows in the queue, DPP can be implemented in conjunction with AFD for the same traffic class. The combination of DPP and AFD provides the best handling of microbursts (aggregated simultaneous mice flows) mixed with a few elephant flows.

DPP uses a flow table to track active flows on ingress ports. If a flow in the flow table remains inactive for longer than the value set for the DPP age-period timer, the flow will be timed out and removed from the flow table. The next time that packets with the same five-tuple arrive at an ingress port, they will be identified as a completely new flow.

## Configuring DPP

### Configuring DPP Parameters

Two parameters are used to configure DPP:

- **age-period:** This parameter defines the time-out timer for an idle flow in the DPP flow table.
- **max-num-pkts:** This parameter defines the maximum number of packets in a mice flow. This number is the threshold value used to distinguish mice flows from elephant flows.

A sample configuration of the DPP parameters on the command-line interface (CLI) is shown here. In this example, the DPP mechanism will send the first 120 packets of each flow to the separate mice-flow queue. After the first 120 packets, excessive packets in a flow will revert to the regular queue for the flow's class in the queueing policy for the egress port. A flow will be timed out if it stays idle for more than 5 milliseconds (ms).

```
hardware qos dynamic-packet-prioritization age-period 5000 usec
hardware qos dynamic-packet-prioritization max-num-pkts 120
```

### Configuring Network-QoS Policy with DPP

DPP is enabled, and its queueing mapping policy is defined in a network-qos policy map that is then applied to the system QoS.

The following example shows a network-qos policy map configured with DPP. In this configuration, DPP is enabled in the default class **c-8q-nq-default on qos-group 0**. Mice flows, or packets of less than the **DPP max-num-pkts** value in each flow, in the default class will be reclassified as belonging to **qos-group 7**. They will be sent to the queue for qos-group 7 on the egress port. The rest of packets, in elephant flows, will be classified as belonging to **qos-group 0**, which is the group for the default class, and will be sent to the default queue on the egress port.

```
DPP Mechanism Parameters:
hardware qos dynamic-packet-prioritization age-period 5000 usec
hardware qos dynamic-packet-prioritization max-num-pkts 120

DPP network-qos policy:
class-map type network-qos c-8q-nq-default
description Default class on qos-group 0
match qos-group 0

policy-map type network-qos dpp
class type network-qos c-8q-nq-default
  dpp set-qos-group 7
  mtu 1500
system qos
service-policy type network-qos dpp
```

In this example, DPP is enabled in the default class. The first 120 packets in each flow in the default class will be sent to qos-group 7. They will go into the queue for qos-group 7 on the egress port.

## Configuring and Applying Queuing Policy with DPP Mice Queue

Continuing the above example, a queuing policy needs to be applied to the egress port to define the priority queuing control in the mice flow queue. In the preceding example, the queue for **qos-group 7, c-out-8q-q7**, is configured as a priority queue in the queuing policy map named **dpp-8q-out**. This queuing policy is applied to an egress port, **Ethernet 1/1**. With this configuration, when a flow reaches the egress port **Ethernet 1/1**, the first 120 packets will be queued in the priority queue **c-out-8q-q7**, and the rest of the flow (an elephant flow) will be sent to the default queue.

```
DPP Mechanism Parameters:
hardware qos dynamic-packet-prioritization age-period 5000 usec
hardware qos dynamic-packet-prioritization max-num-pkts 120

DPP network-qos policy:
class-map type network-qos c-8q-nq-default
  description Default class on qos-group 0
  match qos-group 0

policy-map type network-qos dpp
  class type network-qos c-8q-nq-default
    dpp set-qos-group 7
    mtu 1500
system qos
  service-policy type network-qos dpp

Queueing Policy on Egress Ports:
policy-map type queuing dpp-8q-out
  class type queuing c-out-8q-q7
    priority level 1
  class type queuing c-out-8q-q6
    bandwidth remaining percent 0
  class type queuing c-out-8q-q5
    bandwidth remaining percent 0
  class type queuing c-out-8q-q4
    bandwidth remaining percent 0
  class type queuing c-out-8q-q3
    bandwidth remaining percent 0
  class type queuing c-out-8q-q2
    bandwidth remaining percent 0
  class type queuing c-out-8q-q1
    bandwidth remaining percent 0
  class type queuing c-out-8q-q-default
    bandwidth remaining percent 100

interface Ethernet1/1
  service-policy type queuing output dpp-8q-out
```

When the queuing policy is applied to egress ports, the DPP-reclassified packets (the first 120 packets in flows in the default class) will be sent to the priority queue for qos-group 7 on the egress ports. The rest of the packets in a flow will remain in the default queue.

This is the regular queue for elephant flows in default class.

Optionally, AFD can be enabled in the queuing policy to provide fair bandwidth allocation among elephant flows. The example below modifies the preceding sample configuration by adding AFD to the default queue with the desired queue length of 1 MB.

```
Queueing Policy on Egress Ports:
policy-map type queuing dpp-afd-8q-out
  class type queuing c-out-8q-q7
    priority level 1
  class type queuing c-out-8q-q6
    bandwidth remaining percent 0
  class type queuing c-out-8q-q5
    bandwidth remaining percent 0
  class type queuing c-out-8q-q4
    bandwidth remaining percent 0
  class type queuing c-out-8q-q3
    bandwidth remaining percent 0
  class type queuing c-out-8q-q2
    bandwidth remaining percent 0
  class type queuing c-out-8q-q1
    bandwidth remaining percent 0
  class type queuing c-out-8q-q-default
    afd queue-desired 1 mbytes
    bandwidth remaining percent 100

interface Ethernet1/1
  service-policy type queuing output dpp-afd-8q-out
```

When the priority queuing policy is applied to egress ports, the DPP-reclassified packets (the first 120 packets in flows in the default class) will be sent to the priority queue for qos-group 7 on the egress ports. The rest of packets in a flow will remain in the default queue.

This is the regular queue for elephant flows in the default class.

AFD is enabled in the default queue with a queue length of 1 MB.

## Application Benefits of AFD and DPP

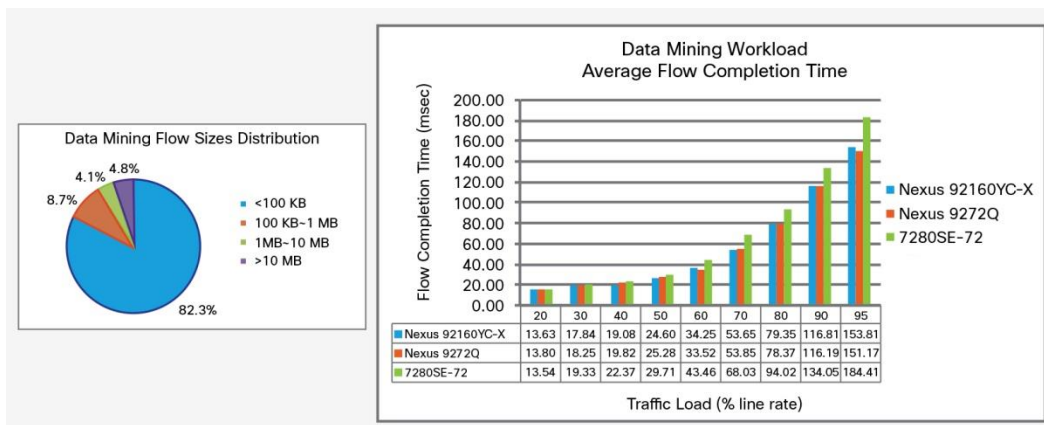
Cisco and a third-party institute have conducted simulation testing, testing with production data center traffic profiles, and testing with real applications to study the impact of AFD and DPP on application performance. These results were compared to application performance results for traditional data center switches with deep buffers but simple buffer management. This section discusses the publicly available testing results listed in references 1 and 2 at the end of this document.

All the applications studied in references 1 and 2 exhibited heavy-tailed flow lengths and traffic load distribution. The tests described in reference 1 studied Hadoop application performance and buffer requirements for a 10 Gigabit Ethernet cluster. These tests compared application performance on a network built with simple deep-buffer switches (represented by Arista 7280SE-72 switches) and application performance on a network with Cisco Nexus 9000 Series Switches with intelligent buffer management. The results showed that the latter provides better performance for the Hadoop application. The results also showed that the Hadoop application does not require deep buffers. The maximum buffer utilization on the simple deep-buffer switch (Arista 7280SE-72) was only about 5 MB per port, whereas it was 1.2 or 1.9 MB on Cisco Nexus 9000 Series Switches.

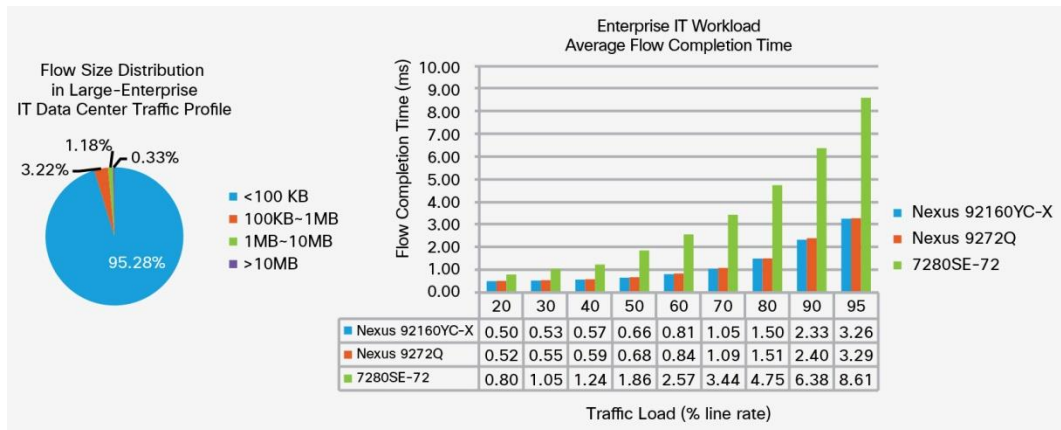
The tests described in reference 2 examined the impact of different buffer structures and management mechanisms on data center application performance. These tests studied two application traffic profiles: for a data mining application and for a typical set of large-enterprise IT data center applications. The flows for these applications were, respectively, 82 percent and 95 percent mice flows of less than 100 KB. In the testing, the same application workloads were launched on two networks under study. One network was built with Cisco Nexus 9000 Series Switches with the intelligent buffer management capabilities. The other network was built with Arista 7280SE-72 switches (representing data center switches with deep buffers but simple buffer management). Application performance was measured based on flow completion times.

The results show that Cisco intelligent buffer management provides significantly better application performance than the simple deep-buffer switch system, particularly when the applications are running at high load. Figures 3 and 4 show the published overall application performance cited in reference 2. For the data mining application, the simple deep-buffer platform had 20 to 30 percent longer flow completion times. For the large-enterprise IT data center applications, Cisco intelligent buffer management provided better flow performance by an order of magnitude.

**Figure 3.** Data Mining Application Flow Distribution and Average Flow Completion Time



**Figure 4.** Large-Enterprise IT Data Center Flow Distribution and Average Flow Completion Time



The switches with traditional simple buffer management treated all flows in a queue equally. They needed to buffer packets in all flows to help ensure that mice flows did not experience packet drops during congestion. The queue grew longer, and all packets in the queue experienced increased queuing latency because they were subject to the same FIFO scheduling. The long queuing latency jeopardized the performance of both mice and elephant flows even though the switches did not drop packets during congestion.

The shorter flow completion time with Cisco Nexus 9000 Series Switches is a direct indicator of improved application performance. Flow completion time was shorter because the Cisco Nexus switches used intelligent buffer management to identify elephant flows and mice flows and apply different queuing management schemes to them that best fit their forwarding requirements. That is, using intelligent management, the switches provided sustained high throughput for elephant flows, with fairness, and they provided sufficient buffer space and low queuing latency for mice flows.

In the testing, both AFD and DPP were enabled on the Cisco Nexus switches. DPP was provisioned so that mice flows on the egress ports were mapped to a low-latency priority queue, helping guarantee low latency for these flows even when the port is temporarily congested as a result of bursts or microbursts. This mapping helped ensure the timely completion of mice flows that carry queries, responses, and quick-synchronization messages. At the same time, AFD worked transparently with the TCP congestion-control mechanism to provide the maximum throughput for elephant flows without overpopulating the egress queue.

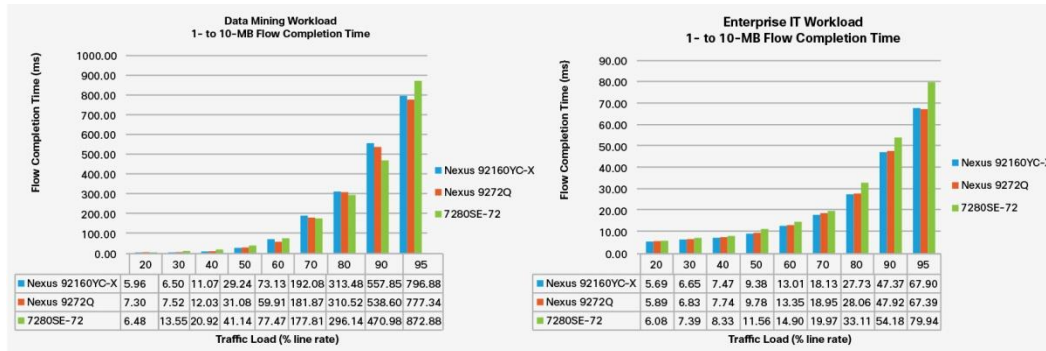
The effects of AFD and DPP on the performance of elephant flows and mice flows, in comparison to the effects of simple buffer management, are analyzed and explained in the following sections.

### Cisco Intelligent Buffer Management with Elephant Flows

In the applications studied, elephant flows constitute about 10 to 20 percent of the flows, but 90 percent of the traffic load. In fact, extensive research shows that these percentages are typical in modern data center networks.

Figure 5 compares the flow completion times of the same elephant flow workloads running on the two networks described in reference 2. With most of the tested workloads, the two networks exhibited similar elephant-flow performance. In some cases, the Cisco Nexus 9000 Series Switches provided better results.

**Figure 5.** 1- to 10-MB Elephant-Flow Completion Time



When the application workload was run through the simple deep-buffer switches, the switches buffered everything to avoid packet drops, with the cost of overgrown queues on the bottleneck link. The longer queue latency undermined the performance benefit of having zero or few packet retransmissions for elephant flows.

When the same workload was sent through Cisco Nexus 9000 Series Switches, AFD actively managed the queue depth by selectively and fairly dropping packets from elephant flows when the queue depth exceeded the specified threshold. The intentional packet drops triggered the TCP congestion-control mechanism on the senders and slowed down packet transmission. Instead of waiting in the queue on the switch, now the excessive data waits on the senders. This approach keeps the switch queue short and the average queue latency low. Although there were more packet drops with AFD than with the simple deep-buffer system, the packet losses were detected and recovered quickly through TCP fast retransmission and had only a minimum impact on flow performance. The benefits of lower queue latency with AFD, however, were significant. As the result, the average elephant-flow completion time remained similar or was better with AFD.

Note that by enabling ECN, it is possible to eliminate the intentional packet drops by AFD. With ECN enabled, AFD will not apply packet drops to elephant flows. Instead, it marks the ECN bit in the packets of elephant flows based on the calculated drop possibility.

Observing and comparing the buffer utilization test results shows that buffer utilization in the two networks was significantly different. Table 2 shows the maximum buffer space used in the tests.

**Table 2.** Maximum Buffer Space Used in Tests

Leaf Switch Tested	Cisco Nexus 9272Q	Cisco Nexus 92160YC-X	Arista 7280E
<b>Documented buffer size</b>	Up to 5 MB per port	Up to 10 MB per port	Up to 50 MB per port
<b>Maximum buffer space in use during testing</b>	2.97 MB	2.95 MB	52 MB
<b>Intelligent buffer capability</b>	Yes	Yes	No

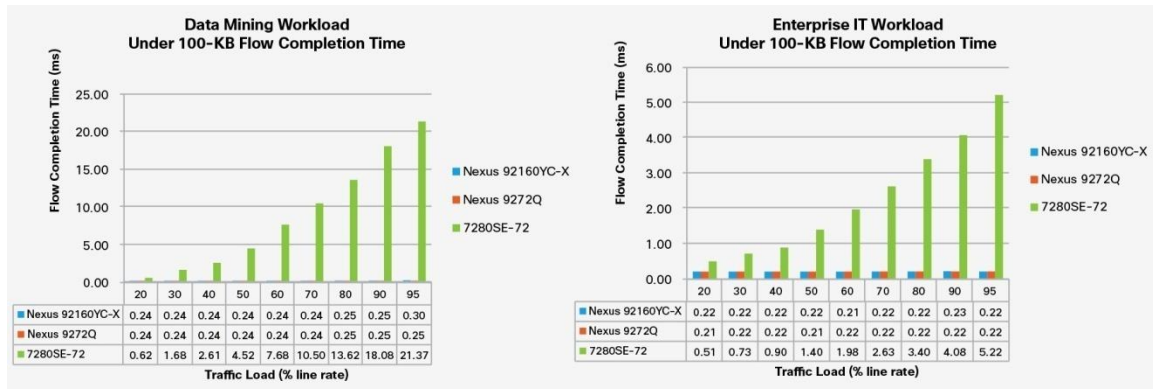
The deep-buffer switch experienced high buffer utilization of 52 MB on the congested port, whereas the maximum buffer utilization on the Cisco Nexus 9272Q and 92160YC-X Switches was less than 3 MB. Memory utilization of 52 MB is about the maximum buffer space available on a port of an Arista 7280E switch. As a consequence of the traditional simple buffer management, the data flows, mostly the elephant flows, occupied all the available buffer space on the port. In comparison, Cisco Nexus 9272Q and 92160YC-X Switches used less than 3 MB of buffer space, only a portion of the available buffer space per port, to handle the same workload and yielded similar or better performance. It is a known fact that adding deep buffer resources to a switch introduces significant extra costs in the making of the switch product. However, as the results show, this additional cost did not provide benefits for the performance of elephant flows in either the data mining application or the typical large-enterprise IT applications.



## Cisco Intelligent Buffer Management with Mice Flows

Mice flows of less than 100 KB constitute about 83 percent of the flows in the data mining application and 95 percent of the flows in the large-enterprise IT applications studied in reference 2. Figure 6 shows the flow completion time for 100-KB mice flows in the testing.

**Figure 6.** Mice Flow (< 100 KB) Completion Time



With both application profiles, Cisco Nexus 9000 Series Switches with intelligent buffer management (Cisco Nexus 92160YC-X and Nexus 9272Q Switches) were used in the testing) and the simple deep-buffer switches (represented by Arista 7280SE-72 in the testing) exhibited very different results for 100-KB mice-flow completion time. On the Cisco switches, the average mice-flow completion time remained in the range of 0.21 to 0.25 ms, independent of the overall traffic load. On the simple-deep-buffer switch, the mice-flow completion time increased as the overall workload increased and reached a maximum of 21.37 ms in the data mining application and 5.22 ms in the enterprise IT applications. Mice-flow performance improved significantly with Cisco intelligent buffer management.

Mice flows are sensitive to latency and packet drops. The simple deep-buffer switches offer enough buffer space to prevent packet drops in mice flows during congestion, but the mice flows have to undergo significantly longer queue latency in the overgrown queue, which prevents them from achieving optimal performance. On the Cisco Nexus 9000 Series Switches with intelligent buffer management, mice flows are protected by AFD and DPP. AFD can reserve adequate buffer headroom for mice flows by limiting the average buffer space that elephant flows can use, which helps ensure that no packet drops occur in mice flows. DPP can be deployed to move mice flows to a separate queue: a low-latency priority queue. This approach allows packets in mice flows to bypass the regular queue and to receive the prioritization and forwarding treatment of a priority LLQ, which leads to the faster completion time for the best performance for mice flows during congestion.

## Conclusion

Modern data center applications have highly distributed architectures with heavy-tailed traffic-flow distribution. Approximately 80 to 90 percent of all flows are mice flows. The other 10 to 20 percent are elephant flows, but these contribute 90 percent of the data traffic load. Elephant flows require high bandwidth and throughput and aggressively occupy buffer space during link congestion. Mice flows are sensitive to latency and packet loss. Aggregated simultaneous mice flows cause microbursts, which are often seen in modern distributed data center applications.

Data center switches with simple buffer management cannot differentiate mice flows and elephant flows. They cannot treat them differently to meet their different forwarding requirements during link congestion. The approach of using simple deep-buffer space to avoid packet drops is expensive, yet it is neither efficient nor effective, trading queuing latency for reduced packet drops. It cannot provide optimal application performance.

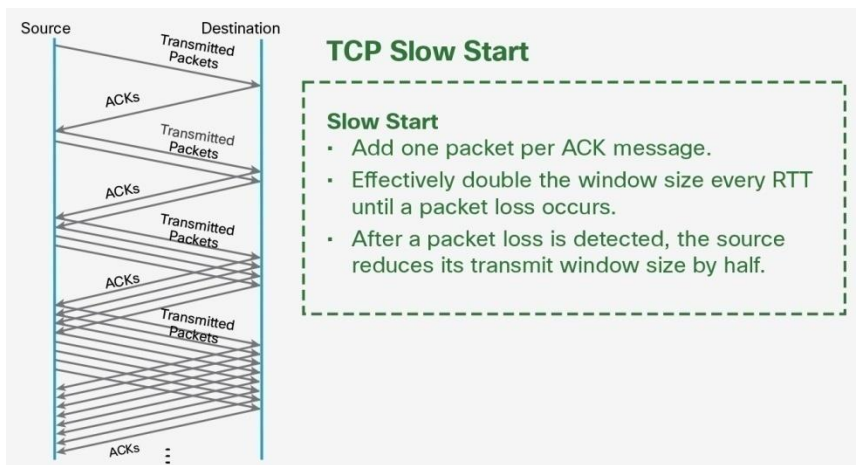
Cisco Nexus 9000 Series Switches with Cisco cloud-scale ASICs introduce intelligent buffer management, including AFD with ETRAP and DPP. With these mechanisms, the Cisco Nexus switches can distinguish mice flows and elephant flows and apply different queue management techniques to them. Mice flows thus can have plenty of buffer space to avoid packet losses during link congestion and can be prioritized in queue scheduling to achieve low latency. At the same time, elephant flows can access the bandwidth they need with fairness based on their data rates. As a result, Cisco Nexus switches with intelligent buffer management functions can provide optimized queue management for both mice flows and elephant flows, leading to optimal flow completion time and better application performance. Cisco's intelligent buffer management addresses the challenges of increased traffic load and heavy incast traffic patterns in the modern data center without the need for excessively large or deep buffer space on the switch. It provides a more effective and cost-efficient way to support modern data center applications, particularly those with microbursts.

## Appendix A: TCP Congestion Control

TCP has its own congestion control mechanism. It is characterized by a slow start followed by a congestion-avoidance phase. The dynamics are governed by the window size (the number of outstanding unacknowledged packets).

A flow starts slowly. The sender increases the window size by 1 each time it receives an acknowledgment (ACK) message. It effectively doubles the packet rate in the next route-trip time, or RTT. This process continues until a packet loss occurs. Figure 7 depicts the slow-start process.

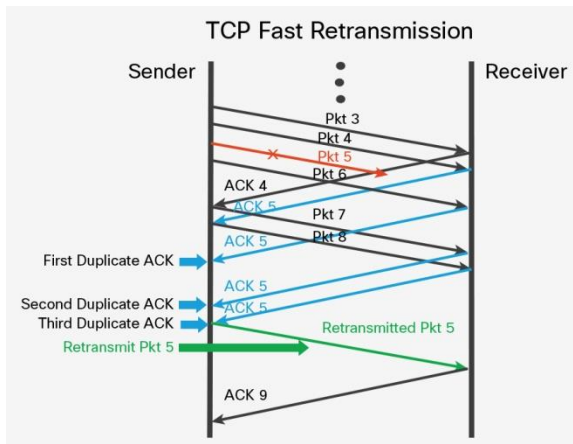
**Figure 7.** TCP Slow Start



TCP uses a packet loss as an implicit congestion notification. To quickly detect and recover from a packet loss to reduce its impact on the entire TCP session, modern TCP implements fast retransmission. Instead of waiting through the TCP retransmission timeout period to confirm a packet loss, fast retransmission, as an enhancement to TCP, detects a packet loss based on duplicated ACK message from the receiver. When the packet of the expected sequence number arrives at the receiver correctly, the receiver acknowledges the packet by sending an ACK message with the sequence number of the next expected packet. When packets arrive out of order, for example, when a packet is missing, the receiver resends the last ACK message to depict the expected packet again. After receiving the third duplicate ACK message, the sender resends the missing packet without waiting for the timer to expire. Upon receipt of the retransmitted packet, the receiver looks at the packets it has received and sends a new ACK message with the expected next packet sequence number.

In the example in Figure 8, the receiver acknowledges packet 4 by sending an ACK message with the next expected sequence number, number 5. Packet 5 gets lost along the way, and packets 6, 7, and 8 arrive at the receiver. Upon receipt of each of them, the receiver sends the ACK message with the expected sequence number 5 again. This behavior causes the sender to receive three duplicate ACK messages with the expected sequence number 5. The sender now knows that packet 5 is lost and will retransmit it. Upon receipt of retransmitted packet 5, the receiver looks through the packets in its receiving window and sends an ACK message with the new expected sequence number. In this example, because packets 6, 7, and 8 have been received in the correct order, the next ACK message will have sequence number 9. On the sender side, in addition to retransmitting the lost packet, the sender considers the packet loss to be a signal of network congestion. It therefore reduces its transmission window by half in an effort to alleviate the congestion.

**Figure 8.** TCP Fast Retransmission



Modern TCP stacks use fast retransmission to quickly detect a packet loss and recover from it by retransmitting the lost packet. In addition to retransmitting the lost packet, the TCP congestion-control mechanism reduces its transmission window size by half to avoid additional congestion. If the previous window size is  $W$ , the new window size becomes  $W/2$ . Because the number of outstanding unacknowledged packets is  $W$ , which exceeds the new window size  $W/2$ , the sender will pause until the number of outstanding packets decreases to  $W/2$ . Then the sender will resume transmission and will gradually increase its window size upon each received ACK message until congestion occurs again: that is, until another packet loss causes the window size to be decreased to  $W/2$  again. When a flow enters this congestion-avoidance phase, its window size, or packet-transmit rate, will exhibit a saw-tooth pattern between  $W$  and  $W/2$ . Figure 9 shows the window size dynamics during this congestion-avoidance phase.

**Figure 9.** TCP Window-Size Dynamics for Congestion Avoidance



As illustrated in Figure 9, TCP uses ACK messages and packet losses as signals to self-clock its transmission of packets. TCP’s saw-tooth congestion-control algorithm is designed to fill any buffer and deliberately cause occasional loss to provide feedback to the sender. No matter how large the buffer at a bottleneck link is, TCP will occasionally overflow the buffer. Therefore, TCP needs packet losses to function well. Packet losses and retransmissions are normal parts of the TCP congestion-control process.

### Appendix B: Sizing Buffers in Networks

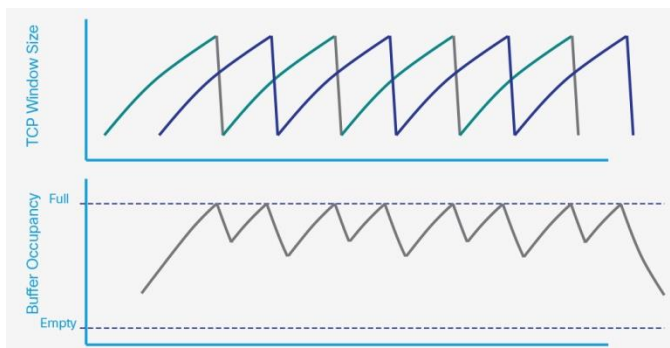
The TCP congestion-control mechanism causes TCP senders to slow down significantly in the event of link congestion and then recover speed gradually after the congestion is resolved. If not enough packets leave the network link during this phase, the link will run idle. That is a waste of the link capacity. To run the link at 100 percent throughput during congestion, the buffer on the network device needs to be large enough to help ensure that there are always packets leaving the link. Extensive studies and experiments have been conducted to determine how to size buffers to sustain 100 percent link bandwidth utilization during congestion. The well-known bandwidth delay product defines the baseline for the buffer size for a single TCP flow:

$$Buffer\ size = C \times RTT$$

Here, C is the link bandwidth, and RTT is the round-trip time for a single flow.

More recent studies have focused on buffer sizing with multiple data flows in the network, which is a more realistic scenario in production networks. These new findings show that the buffer size needed for 100 percent link utilization with multiple flows is much smaller than the size determined by the bandwidth delay product because the multiplexed flows smooth out the burstiness of traffic during congestion, as shown in Figure 10.

**Figure 10.** Multiplexed TCP Flows



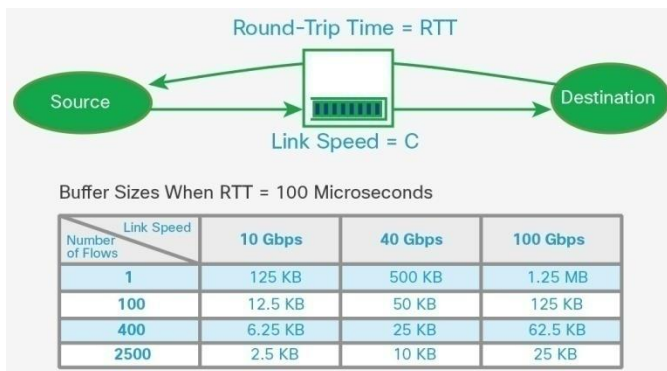
The needed buffer size is the bandwidth delay product value divided by the square root of the number of flows:

$$Buffer\ size = \frac{C \times RTT}{\sqrt{N}}$$

Here, C is the link bandwidth, RTT is round-trip time, and N is the number of long-lived flows (see reference 6 at the end of this document).

Using an average RTT of 100 microseconds in a data center network, Figure 11 shows the buffer size for different link speeds and various numbers of flows. Note that the buffer size decreases rapidly as the number of flows increases. For instance, on a 100-Gbps link with 2500 flows, only a 25-KB buffer is needed.

**Figure 11.** Buffer Sizing for Different Link Speeds and Numbers of Flows



After the buffer contains enough packets to keep the congested link running at 100 percent throughput, any additional buffer space will not provide any additional throughput benefit. However, because TCP’s sawtooth congestion-control algorithm is designed to fill any buffer space and deliberately causes occasional packet loss to provide congestion feedback to the sender, the additional buffer space will be filled, causing the queue latency to increase. Long queuing latency adversely affects overall application performance. Therefore, choosing the right buffer size and the right buffer management scheme is important to enable the network to provide optimal performance and support for applications.

## References

1. Network Switch Impact on “Big Data” Hadoop-Cluster Data Processing (<http://miercom.com/network-switch-impact-on-big-data-hadoop-cluster-data-processing/>)
2. Cisco Systems Speeding Applications in Data Center Networks (<http://miercom.com/cisco-systems-speeding-applications-in-data-center-networks/>)
3. M. Alizadeh et al., Data Center TCP (DCTCP), SIGCOMM 2010
4. M. Alizadeh et al., CONGA: Distributed Congestion-Aware Load Balancing for Data Centers, SIGCOMM 2014
5. Greenberg et al., VL2: A Scalable and flexible Datacenter Network, SIGCOMM 2009
6. G. Appenzeller et al., Sizing Router Buffers, SIGCOMM 2004




---

**Americas Headquarters**  
Cisco Systems, Inc.  
San Jose, CA

**Asia Pacific Headquarters**  
Cisco Systems (USA) Pte. Ltd.  
Singapore

**Europe Headquarters**  
Cisco Systems International BV Amsterdam,  
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at [www.cisco.com/go/offices](http://www.cisco.com/go/offices).

 Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: [www.cisco.com/go/trademarks](http://www.cisco.com/go/trademarks). Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)