

Solaris™ 9 Network Resource Management using IPQoS

Technical White Paper
May 2003



Table of Contents

Introduction	1
Workload Management	2
Intended Audience	2
Differentiated Services	3
Per-Hop Behaviors (PHBs)	5
Rate Measuring: Metering	6
Congestion Management: Queuing and Dropping	7
Rate Enforcing: Shaping and Policing	8
802.1D Priority Marking	9
Solaris 9 Network Resource Management	11
Architecture	12
Packet Classification	13
Metering	14
Marking	14
Monitoring	15
Network Accounting	16
Solaris Bandwidth Manager	18
Feature Comparison Between Solaris 9 IPQoS and Solaris Bandwidth Manager	20
IP Type of Service (ToS) Application APIs	22
Solaris Socket Library	22
Java Technology	23

Chapter 1

Introduction

The Solaris™ 9 9/02 Operating System release introduces new Network Resource Management technology, superseding the Solaris Bandwidth Manager product available for previous Solaris releases. This technology is usually referred to as IP quality of service (QoS), or simply IPQoS.

Solaris 9 Network Resource Management technology is ideal for:

- Controlling network resources
- Monitoring network resources
- Accounting network resources

Solaris 9 Network Resource Management technology achieves these goals by:

- Standards compliance to, and interoperability with, IETF Differentiated Services standards and specifications
- Integration with other Solaris resource management technologies in the area of monitoring, accounting, and workload management

These features can be used independent of each other. Network resource monitoring for example can be very useful, even without specifically controlling these resources.

Workload Management

An often heard misconception about network QoS technology is that it is thought to be only useful in areas where network bandwidth is very limited and/or expensive. However, network resource management technology can be extremely useful in the data center, even when network bandwidth is plentiful. It supports the move towards data center virtualization by aiding in the management of discrete classes of network traffic. This helps to enable workload-centric data center management, as opposed to hardware-centric management.

Managing the usage of network resources enables business processes such as capacity planning, chargeback, server/network consolidation, and workload prioritization. And the benefits are plentiful. For example, new services could be deployed on the existing hardware infrastructure, without requiring up-front hardware investments. Users can be charged for the exact amount of network resources that have been consumed. Resources can be assigned dynamically to the workloads that will generate the most business ROI.

Solaris 9 IPQoS technologies are linked with other Solaris resource management technologies. For a overview of Solaris 9 Resource Management, see the *Solaris 9 Resource Manager Software* technical white paper.

Managing workloads instead of hardware is also the fundamental concept implemented by Solaris Containers. For more information about this, please see the *Solaris Containers — How Advances in Server Virtualization Will Simplify Service Manageability* white paper. Solaris 9 IPQoS forms the foundation of managing the network resource aspects of Solaris Containers.

Intended Audience

This technical white paper is intended for IT managers, architects, and system administrators who need a better understanding of Solaris 9 Network Resource Management technology. For customers already familiar with Solaris Bandwidth Manager software, the key differences will be highlighted.

No detailed knowledge of quality of service concepts or technology is assumed. Chapter 2 introduces the reader to the DiffServ architecture, and many applicable QoS concepts. Chapter 3 outlines how these concepts are implemented in the Solaris 9 Operating System. Chapter 4 briefly outlines Solaris Bandwidth Manager software, for comparison purposes.

Chapter 2

Differentiated Services

The Internet Engineering Task Force (IETF) has created several standards drafts and specifications for class-based traffic differentiation, by classifying and marking traffic aggregates. This is referred to as Differentiated Services, or simply DiffServ. Basically, the idea behind DiffServ is to give certain traffic aggregates different packet-forwarding treatment than others. Examples of such aggregates could be all voice-over-IP traffic, or financial data at the end of each fiscal quarter, or all network traffic going to and from the company chairman's laptop.

The IETF DiffServ charter can be found at ietf.org. The main RFCs that describe DiffServ are:

1. RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers
2. RFC 2475: An Architecture for Differentiated Services

RFC 2474 redefines the use of the type of service (ToS) byte (8-bit field) in the IP header, and now refers to this header field as the DS codepoint, or DSCP (see Figure 2-1). By marking all IP packets belonging to an aggregate with a specific DSCP, other nodes in the network (if also DiffServ aware) can give end-to-end differential treatment to this aggregate based on the DSCP found in each packet.

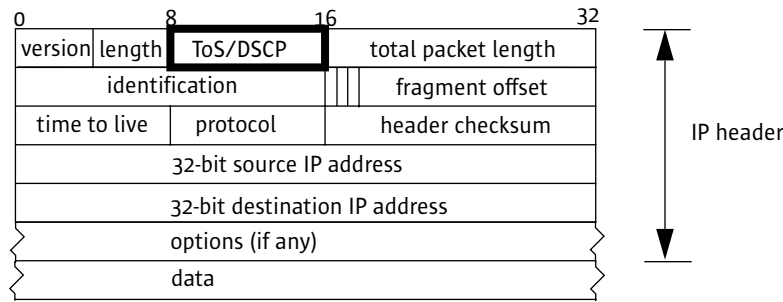


Figure 2-1: 8-bit Type of Service/Differentiated Services Codepoint in an IP Header

Only the first 6 bits are used for marking the DSCP. The last 2 bits are reserved for other use (explicit congestion notification, or ECN (see Figure 2-2)).

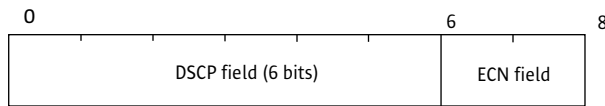


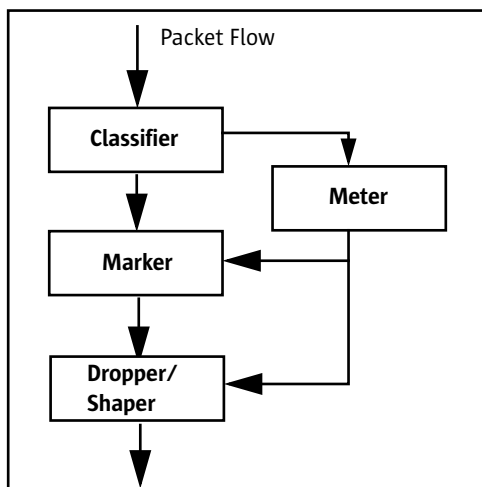
Figure 2-2: Differentiated Services Codepoint Format

The Differentiated Services architecture (RFC 2475) describes the following functional components (see Figure 2-3):

The classifier aggregates packet flows into discrete classes (aggregates).

- Meters can be used to measure conformance of aggregates against predefined profiles (e.g., a data rate, or more broadly, a service level).
- Markers are used to mark the packets' DiffServ codepoint, based on the result of the metering/classification function.
- Droppers will drop out of profile packets.
- Shapers will streamline packets into profile conformance by using various queuing and/or other congestion avoidance techniques.

Figure 2-3: DiffServ Functional Components



Different combinations of meters, markers, and droppers/shapers can be applied in an implementation. Together, this functionality is also referred to as traffic conditioning.

Per-Hop Behaviors (PHBs)

In DiffServ-enabled networks, when multiple interconnected nodes agree on compatible Per-hop Behaviors, aggregations of flows can get consistent, differentiated levels of service. According to RFC 2475 (An Architecture for Differentiated Services), “a per-hop behavior (PHB) is a description of the externally observable forwarding behavior of a DS node applied to a particular DS behavior aggregate.” Different mechanisms can be used to implement specific PHB behaviors. Devices on the edge of a DiffServ domain classify a packet by looking at various properties, such as packet header information. They then mark the packet with a DiffServ codepoint, indicating the desired PHB behavior. The main benefit here is that network devices in the core of the network only have to examine a packet’s DSCP in order to determine its forwarding behavior.

The IETF currently has defined the following Per-hop Behaviors:

- RFC 2597: Assured Forwarding PHB Group
- RFC 2598: An Expedited Forwarding PHB¹

Assured Forwarding

The Assured Forwarding (AF) PHB group offers multiple forwarding assurances. It currently defines four independent forwarding classes, each with its own resources (such as buffers and/or bandwidth) available to it. Each of the four classes defines three separate drop precedences. If congestion occurs within a class, the drop precedence determines the likelihood that packets get discarded or queued. Together, the four classes and three drop precedences form twelve AF codepoints, indicated as AF_{xy}, where x is the class number and y is the precedence level (See Figure 2-4). The recommended DiffServ codepoints are included in 1 for reference as well.

Table 2-4: Assured Forwarding Classes and Recommended Codepoints

	Class 1	Class 2	Class 3	Class 4
Low Drop Precedence	AF11 (001010)	AF21 (010010)	AF31 (011010)	AF41 (100010)
Medium Drop Precedence	AF12 (001100)	AF22 (010100)	AF32 (011100)	AF42 (100100)
High Drop Precedence	AF13 (001110)	AF23 (010110)	AF33 (011110)	AF43 (100110)

Expedited Forwarding

The Expedited Forwarding (EF) PHB can be used to build low-loss, low-latency, low-jitter, assured-bandwidth, end-to-end services through DiffServ domains. In order to achieve this, packet queuing must be avoided (or at least minimized) at each hop along the packet’s path. Nodes that implement the EF PHB should support a configurable minimum rate for EF aggregates, which limits the effect that non-EF traffic can have on the EF rate. On the other hand, EF traffic must not overwhelm and starve non-EF traffic either, a maximum rate (also called burst rate) must be configurable. EF implementations can minimize queuing by ensuring that an aggregate’s arrival rate is not greater than its departure rate on each node. Arrival rates can be controlled using policing or shaping techniques. The DSCP value 101110 is recommended for the EF PHB.

1. There are also some more recent variants or improvements to the Expedited Forwarding PHB, described in other RFC documents. See the IETF web site for the latest developments.

Rate Measuring: Metering

Meters are used in DiffServ to measure packet streams. The meter is an important functional element of the DiffServ architecture. A meter can determine whether a packet stream is flowing at a rate higher than the allowed peak rate. It can take action based on the meter's result, such as dropping the packet, or marking it with a high drop precedence codepoint.

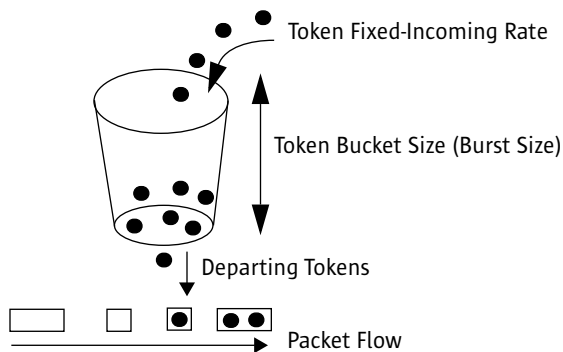
Since there is not a single “right” way to measure packet streams, the DiffServ architecture does not dictate particular implementation. A meter is just a mechanism that can help achieve a desired behavior. How exactly this behavior is achieved is less important.

Several informational and experimental RFCs have been proposed that specify different metering mechanisms that can be used in product implementations. Examples are:

- RFC 2697 (Informational): A Single Rate Three Color Marker
- RFC 2698 (Informational): A Two Rate Three Color Marker
- RFC 2859 (Experimental): A Time Sliding Window Three Color Marker (TSWTCM)

Meters can be designed with specific Per-hop Behaviors in mind. What the meter RFCs mentioned above have in common is that their output is one of the three colors green, yellow, or red.

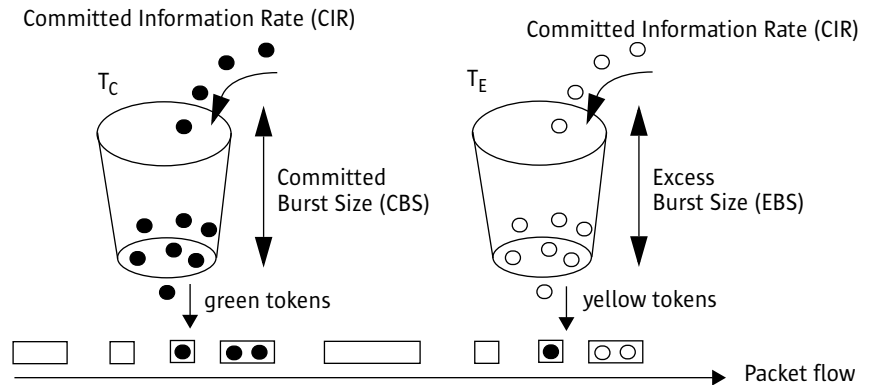
Figure 2-5: A Token Bucket



Token Bucket Meters

In both the Single Rate Three Color Marker and the Two Rate Three Color Marker, token buckets are used to implement their function. Both of these token bucket meters can be configured in so-called color-blind mode, as well as color-aware mode. In both cases, a token bucket gets filled at a steady, predetermined rate. This is the rate that packet streams are measured against. When the bucket is full, any “overflowing” tokens are lost. Tokens are removed from the bucket and assigned to packets in accordance to their size. See Figure 2-5.

Figure 2-6: Single Rate Three Color Marker



The Single Rate Three Color Marker uses two token buckets T_C and T_E , which each get filled at the steady rate of CIR tokens (bytes) per second. Each token bucket has its own (burst) size. When packets arrive, tokens corresponding to the packet's size have to be found. If tokens are available in T_C , they are taken out of that bucket, and the packet is considered green. If not enough tokens are present in T_C , but are available in T_E , then the packet is considered yellow. If neither of the buckets has enough tokens available, then the packet is considered red.

The Two Rate Three Color Marker is a variation of this, where the second token bucket has its own independent rate called the peak rate. Tokens/colors are assigned in a slightly different manner. If not enough tokens are available in the peak bucket, the packet is red. If not enough tokens are available in the committed rate bucket, the packet is yellow. The packet is green and corresponding tokens are taken from both buckets.

Color-aware mode differs from color-blind mode in that in color-aware mode, the preexisting color of a packet is taken into account when assigning a new color.

The Time Sliding Window Three Color Marker doesn't use token buckets, but instead uses an algorithm to estimate the packet stream's rate. This rate estimator utilizes a time decay factor to deal with fluctuation in packet rates over time, hence the term time sliding window. Based on observations over time, a rate estimate is determined. This rate estimate is compared against two configured rates: the Committed Target Rate (CTR) and the Peak Target Rate (PTR). If the estimated rate is lower than the CTR, the packet is green. If it is between the CTR and PTR, the packet is yellow. If the estimated rate is higher than the PTR, the packet is red.

A packet's color can determine its forwarding behavior. For example, when Assured Forwarding is being used, green packets could get marked low drop precedence, yellow packets marked medium drop precedence, and red packets marked high drop precedence. When congestion occurs elsewhere in the network, high drop precedence packets then could get dropped first.

Congestion Management: Queuing and Dropping

If there was always sufficient resources available in the network to process and forward packets without any delay, optimal quality of service would always be achieved. However, even in the best network topologies, queuing and dropping are mechanisms being utilized all the time.

For example, take a scenario where a server is connected to a gigabit Ethernet port on a switch, and a client connected to a fast Ethernet port on the same switch. If the server sends data at full speed to the client, the switch will run out of resources on the client's port very quick, and start dropping packets. If the server uses TCP (or flow control at the application level in case UDP is being used), it will back down and end up sending at a rate roughly equal to the fast Ethernet's line speed rate. The switch also could have a number of buffers available (i.e., a queue) to store packet's that cannot be forwarded directly. If there is a short burst of network traffic from the server, the switch might be able to absorb the burst in its queue without having to drop any packets.

In many environments, the network is a medium that is shared between its users. Problems can arise when multiple users contend for the same resources at the same time. This could lead to undesirable situations, for example, when one user is a video conferencing application and another user is a backup application. The video conferencing application could start experiencing undesirable jitter from lost or queued packets, caused by the resource-hungry network backup job. This effect can be mitigated by dedicating resources to specific workloads across the network. Instead of randomly dropping packets when congestion occurs, perhaps only backup traffic could get dropped. Or maybe if the video conferencing application could get its own queue, and would get a higher priority in getting packets processed from this queue.

DiffServ enabled devices often have the capability to do just that. The relevant features include:

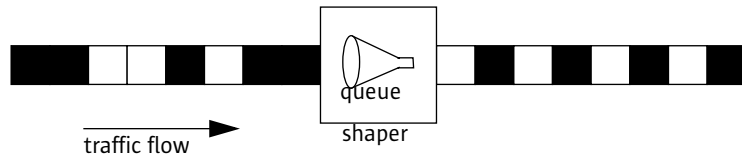
- A queuing/scheduling mechanism. Queues can be implemented in hardware, in software, or using a combination of the two. Routers and switches often have a limited number per-port hardware queues available. The scheduling algorithm determines in which order packets are processed from the queues. Examples of better-known queuing/scheduling algorithms are Round Robin, Weighted Round Robin, or Class-Based Queueing.
- A packet dropping mechanism. When congestion occurs within a class, typically a queue will fill up, When the queue is full, new packets will simply get dropped. This is also called tail-dropping. More sophisticated mechanisms exist to drop packets, such as Random Early Detection (RED) or Weighted Random Early Detection (WRED). In both cases, packets will get dropped randomly (and semi-randomly, respectively) from anywhere in the queue instead of tail-dropping, after a certain queue depth is reached. The goal is to avoid heavy congestion by proactively attempting to slow down senders before the queue is completely full.

Rate Enforcing: Shaping and Policing

As discussed previously, meters measure a packet stream's rate and determine whether the rate is within configured limits. A packet stream is said to be in-profile if its rate falls within a limit, and out-of-profile if the measured rate exceeds the configured rate. Actions can be taken depending on whether packets are in-profile or out-of-profile, such as marking an out-of-profile packet with a higher drop precedence. A device could also take direct action to slow down out-of-profile packet streams and enforce the configured rate. Two relevant techniques for this are traffic shaping and traffic policing.

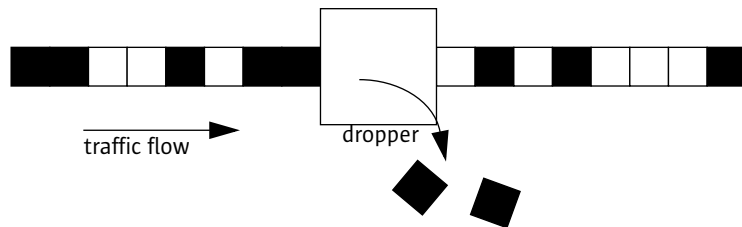
Traffic shaping works by queuing out-of-profile packets and scheduling packets from this queue at the configured rate. This technique can help smooth bursty packet streams, and help remove jitter effects. Figure 2-7 illustrates this.

Figure 2-7: Traffic Shaping



Traffic policing works by simply disregarding (dropping) out-of-profile traffic. It won't attempt to buffer the packet for later scheduling. See Figure 2-8.

Figure 2-8: Traffic Policing



There is a very fine line between packet dropping as a result of congestion (avoidance) and rate limiting (policing) a packet stream. Both involve dropping packets, and the same mechanism could be used to implement them.

802.1D Priority Marking

DiffServ works by aggregating and marking packet streams at the IP (layer 3) level. The IP header contains the DiffServ codepoint. Routers process IP headers, and can act on the DiffServ codepoints when needed.

There is value in also establishing differentiated classes of service at the data link layer. In a “switched” layer 2 environment, for example, within a data center, multiple workloads might require differentiated treatment.

Layer 2 LAN protocols are governed by the IEEE 802 committee. The committee consists of many sub committees dealing with LAN-specific characteristics, such as 802.3 (CSMA/CD — “Ethernet”), 802.4 (token bus), and 802.5 (token ring). The 802.1 committee deals with things common to all of these, including MAC headers and addressing, spanning trees, etc. Within the 802.1 committee, there are several standards defined such as 802.1D for MAC bridging, 802.1Q for VLANs, etc. MAC addressing issues are defined in the 802.1D standard, and hence apply to all 802 LANs.

The original MAC header format did not have a header field defined for priority, similar to the IP type of service header field. But to make VLANs work, a VLAN identifier needed to fit in the MAC header. However, since there was no space in the MAC header for this either, the 802.1Q working group extended the MAC header format. In addition to a VLAN identifier field, this new header format also included a 3-bit traffic class field. Another standard, 802.1p, dealt with the specifics of this 3-bit traffic class field, and defined the 8 possible traffic classes.

The 802.1D standard has since subsumed the 802.1Q and 802.1p standards. People often still refer to 802.1p when they really mean 802.1D traffic classes. And because of Ethernet’s overwhelming popularity, people also simply refer to “Ethernet class of service (CoS).”

In order to take advantage of 802.1D traffic classes, VLANs must be enabled on those network segments. Many network topologies today implement port-based VLANs, which typically do not involve the extended MAC header because the network device simply associates all network traffic to that port with that VLAN. Therefore, a VLAN identifier is not needed and generic MAC headers can be used.

The 802.1D standard defines the following traffic classes:

Value	Name
0	Background
1	Default
2	Best Effort
3	Excellent Effort
4	Controlled Load
5	Video
6	Voice
7	Network Control

Table 2-9: 802.1D traffic classes

There are currently no specific standards that describe the relationship between 802.1D traffic classes and DiffServ Per-hop Behaviors. Sometimes, devices are capable of mapping DiffServ codepoints to specific 802.1D priorities. However, this mapping is not as trivial as it might seem. First, it depends on the actual implementation. Does the device have dedicated resources for each priority level? Perhaps a layer 2 switch only has two queues per port, and maps priorities 0–3 to one queue, and priorities 4–7 to the other queue. Second, 802.1D traffic classes mix business criticality with workload characterization. Should a video conferencing application that is not mission critical get better treatment than a mission-critical ERP application?

Chapter 3

Solaris 9 Network Resource Management

Solaris 9 Network Resource Management technology (also known as Solaris 9 IPQoS) supercedes Solaris Bandwidth Manager software. IPQoS was first made available in the Solaris 9 9/02 release. It is fully integrated in the IP layer of the Solaris TCP/IP networking stack.

Solaris 9 Network Resource Management technology has features for controlling, monitoring, and accounting of network resources. The control features are DiffServ compliant for optimal integration with QoS-enabled networks; the monitoring features are compatible with the standard Solaris statistics framework (kstat), and the accounting features are compatible with other Solaris resource accounting methods (using the Solaris extended accounting framework).

The software provides an implementation that can be used to deploy the features described in the DiffServ RFCs 2474 and 2475. It supports the Assured Forwarding (RFC 2597) and Expedited Forwarding (RFC 2598) Per-hop Behaviors, and in addition to marking DiffServ codepoints, can also mark VLAN-enabled, layer 2 Ethernet class of service (CoS) header fields (IEEE 802.1D/802.1p).

Architecture

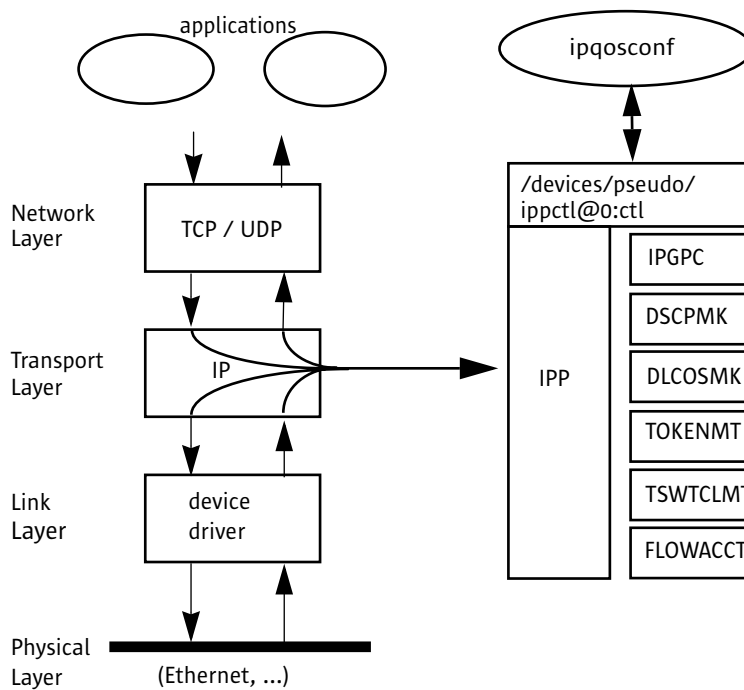
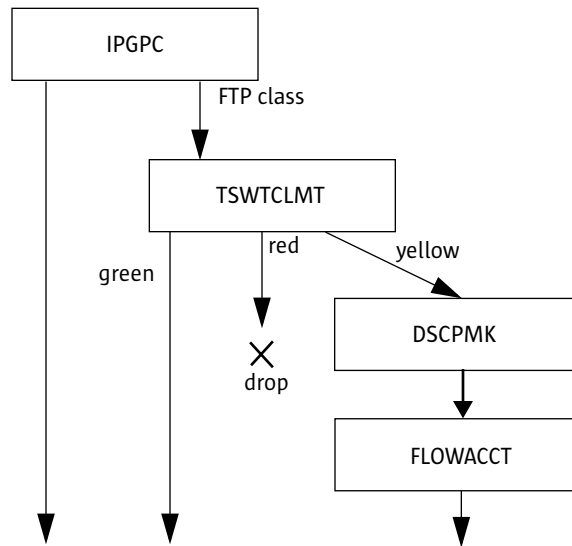


Figure 3-1: Solaris 9 Network Resource Management Architecture

Central to the Solaris 9 Network Resource Management architecture is the IP Policy Framework (see Figure 3-1). This is a modular framework with callback hooks from and to the IP module. A generic packet classifier is implemented as a module in the IP Policy Framework, as well as a number of actions. Actions are executed as a result of the classifier. Multiple actions can be chained together. Actions relevant to Solaris 9 Network Resource Management are dropping, metering, marking, or flow accounting.

Configuration tasks are accomplished through the `ipqosconf(1M)` utility, which uses a pseudo device driver to access the modules. The desired functionality is obtained by chaining modules in a particular order. The first module in the chain is always the packet classifier. The packet classifier assigns packets to classes, and for each class, one or more actions can be invoked serially. An example module chain is shown in Figure 3-2.

Figure 3-2: IPQoS Action Chain



In this example, the packet classifier is invoked and classifies FTP traffic in a class. All traffic for this class is sent to the Time Sliding Window Three Color Marker meter (TSWTCLMT). In the example, red-colored traffic is dropped, and yellow-colored traffic is sent to the DSCP marker module, where it can be marked with, for example, an Assured Forwarding DSCP codepoint. After DSCP marking, the flow accounting module is invoked to update an accounting record for this flow.

Packet Classification

The Solaris 9 IP packet classifier is a multifield (MF) packet classifier, and can classify packets based on one or more of the following selectors:

- Source/destination IP address and mask
- Source/destination port number
- Protocol
- IP version (IPv4, IPv6)
- Direction (local in, local out, fwd in, fwd out)
- Interface name, interface group name
- TOS/DSCP value
- User ID (uid); local outgoing only
- Project ID (projid); local outgoing only

The administrator creates filters using one or more of these selectors, and based on these filters, packets are associated with a class. Each class then has its own traffic conditioning behavior, depending on the configuration. Multiple filters can be associated with a single class.

Filters can overlap each other, meaning that a packet might match more than one of the configured filters. When this is the case, only one filter will apply. The classifier allows packets that match multiple filters to behave in a deterministic way by explicitly assigning a precedence value to overlapping filters in the configuration.

User ID and project ID filtering allows packets belonging to a process' user ID or project ID. Currently, this only works for outgoing packets. Integration with the projects administrative framework allows for easy integration with other Solaris 9 Resource Management technologies. For more information about projects, please see the *Solaris 9 Resource Manager Software* technical white paper. Outgoing packets are classified before fragmentation or IPSec encryption. Incoming packets are classified after reordering, reassembly, or IPSec decryption.

Note – Due to the nature of packet classification, kernel compute resources are required for inspecting all network packets. The exact amount of this overhead cannot easily be characterized because it is very much dependent on the specific IPQoS configuration, as well as the nature of the application workload(s).

Metering

Three different meters are currently implemented:

- RFC 2697: A Single Rate Three Color Marker
- RFC 2698: A Two Rate Three Color Marker
- RFC 2859: A Time Sliding Window Three Color Marker (TSWTCM)

Each of these meters determines a color for each packet being measured. Based on the color, any action can be invoked. For example, DSCP marking could be done based on corresponding Assured Forwarding drop precedence codepoints. Or, the dropper could be invoked when the color is red, and the flow accounting module is invoked when the color is yellow. See Figure 3-2 for an example in graphical format.

The Single Rate Three Color Marker and Two Rate Three Color Marker are implemented in a single module called “tokenmt.” Depending on whether or not a peak rate is being configured, the module will switch between the single-rate or two-rate meter behavior. Both of these meters support color-blind as well as color-aware modes.

The Time Sliding Window Three Color Marker is implemented in a module called “tswtcmt.” Each meter exports run-time statistics using the kstat facility. These statistics include how many green, yellow, and red bits and packets have occurred per time slot for each instance of each meter.

Marking

Two marker modules are implemented. The dscpmk marker module is used to mark the DSCP codepoint in the IP header. The dlcosmk (Data Link Class of Service) marker module can be used to mark the 802.1p priority field in the 802.1Q VLAN extended Ethernet header of supported VLAN-aware Ethernet devices. (Currently, only the GigaSwift gigabit Ethernet adapter is supported.)

The markers can re-mark packets where appropriate. For example, applications can set the ToS on the socket with a certain value, which can be re-marked by the dscpmk marker if needed.

Monitoring

Each module in the IP Policy Framework can export its own run-time statistics using the standard Solaris kstat facility. This integration enables administrators to use kstat to start monitoring workloads, in addition to infrastructure (the network interfaces themselves).

The `ipgpc` classifier module can export global statistics as well as per class statistics:

```
% kstat -m ipgpc
module: ipgpc                instance: 6
name:  ipgpc_global_stats    class:  ipgpc.classify
      crtime                 1158291.77422662
      epackets                0
      nbytes                  1273772
      nclasses                4
      nfilters                3
      npackets                4745
      snaptime                1162042.47026118
module: ipgpc                instance: 6
name:  c_oracle              class:  ipgpc.classify
      crtime                 1158291.7784389
      last_match              1160722411595091
      nbytes                  1080
      npackets                2
      snaptime                1162042.46604117
```

In the example `kstat(1M)` output above, `c_oracle` is a class name given by the system administrator using `ipqosconf(1M)`. Per-class statistics are an excellent way to get workload-specific, networking point-in-time statistics. This is useful for example for monitoring, debugging, or capacity planning purposes.

The meter modules exports statistics indicating packet rates in its respective color buckets:

```
% kstat -m tokenmt
module: tokenmt              instance: 7
name:  tokenmt_statistics    class meter5mbps
      crtime                 1167240.92663574
      epackets                0
      green_bits              167104992
      green_packets           28743
      red_bits                 0
      red_packets             0
      snaptime                1180483.22191483
      yellow_bits              26794560
      yellow_packets          2264
```

Similarly, the marker modules export the following:

```
% kstat -m dscpmk
module: dscpmk               instance: 11
name:  dscpmk_stats          class:  af11
      crtime                 1167240.92492631
      dscp_changed           24
      dscp_unchanged         0
      epackets                0
      ipackets                25
      npackets                49
      snaptime                1174389.99876677
```

Depending on the actual configuration, multiple `kstat` instances of a given module can exist.

Network Accounting

Network flow accounting records how much networking resource was used, by whom, and when. Accounting records can be written out using the Solaris extended accounting framework, similar to other Solaris 9 resource management technologies.

For each network flow belonging to a class, the following properties can be written and extracted into records:

- Source IP address (IPv4 and IPv6 supported)
- Destination IP address (IPv4 and IPv6 supported)
- Source port
- Destination port
- Protocol
- ToS/DiffServ codepoint
- Number of bytes
- Number of packets
- Action name which invoked this accounting record
- Flow creation timestamp
- Flow last seen timestamp
- Project ID of the process (when available; outgoing flows only)
- User ID of the process (when available; outgoing flows only)

The Solaris extended accounting framework can be configured using `acctadm(1M)` to write any of these flow properties into each flow accounting record. Two optional predefined templates — basic and extended — can be used and contain the following properties:

```
% acctadm -r flow
extended  saddr, daddr, sport, dport, proto, dsfield, nbytes,
npkts, action, ctime, lseen, projid, uid
basic     saddr, daddr, sport, dport, proto, nbytes, npkts, action
%
```

The extended accounting feature for flows can be turned on using `acctadm(1M)` as follows:

```
# acctadm
    Task accounting: inactive
    Task accounting file: none
    Tracked task resources: none
    Untracked task resources: extended
    Process accounting: active
    Process accounting file: /var/adm/processacct
    Tracked process resources: basic
    Untracked process resources:
projid,taskid,host,mstate,ancpid,wait-status
    Flow accounting: inactive
    Flow accounting file: none
    Tracked flow resources: none
    Untracked flow resources: extended
# acctadm -e basic,ctime,lseen -f /var/adm/flowacct flow
# acctadm
    Task accounting: inactive
    Task accounting file: none
    Tracked task resources: none
    Untracked task resources: extended
    Process accounting: active
    Process accounting file: /var/adm/processacct
    Tracked process resources: basic
    Untracked process resources:
projid,taskid,host,mstate,ancpid,wait-status
    Flow accounting: active
    Flow accounting file: /var/adm/flowacct
    Tracked flow resources: basic,ctime,lseen
    Untracked flow resources: dsfield,projid,uid
#
```

Flow accounting records are unidirectional. Each record only captures data travelling from the source IP address to the destination address. The reverse traffic can be captured by enabling a corresponding accounting record in the reverse direction, and matching the records with corresponding connection identifiers (source/destination IP addresses and port numbers).

The Solaris kernel keeps track of flow records that are in progress and have not been written out yet. This can be monitored using `kstat(1M)` on the `flowacct` module. A configurable timeout parameter enables records to be written out before connections have expired. If more packets belonging to that connection arrive after the timeout has expired, a new accounting record will be created for these new packets belonging to that flow. Long-lived connections, therefore, can be represented in multiple flow accounting records.

The flow accounting records can be extracted using the Solaris `libexacct(3LIB)` API. A Perl API is planned to be made available in the future. Third party applications enabling features such as chargeback, billing, or capacity planning using flow accounting are expected to become available also.

Chapter 4

Solaris Bandwidth Manager

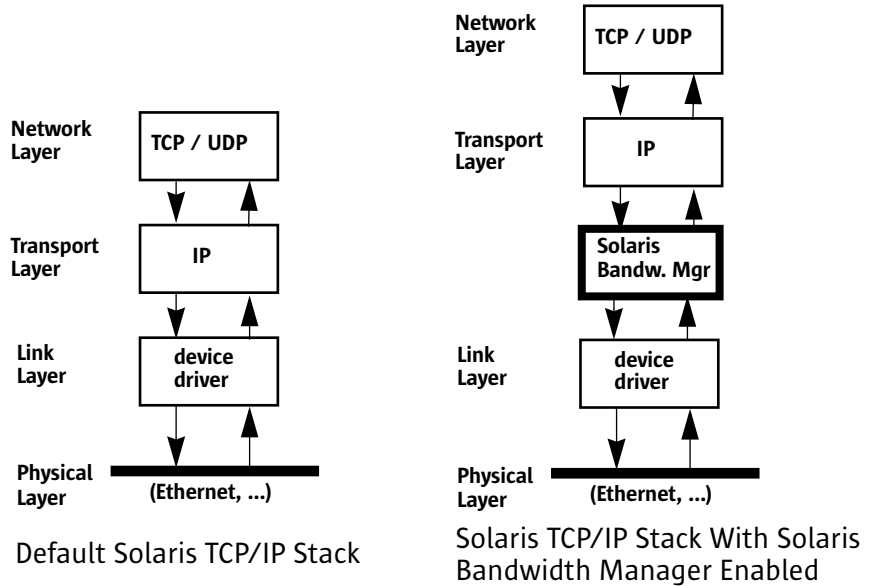
Solaris Bandwidth Manager software was designed to provide customers with the tools to QoS-enable their networks. The product predates many of the more recent standards work of the IETF around DiffServ, and therefore lacks implementations of specific PHBs or meters. However, the product does support DiffServ by being able to classify and mark packets with any ToS/DSCP codepoints. It also supports packet shaping and policing using a Class-Based Queuing (CBC) implementation.

Some key characteristics and features of Solaris Bandwidth Manager include:

- Layered design
- Support for “IP-transparent mode” allowing a host running Solaris Bandwidth Manager software to be deployed in front of a legacy (with no support for QoS) router
- URL filtering, enabling bandwidth allocation to specific URLs
- Rich, custom APIs for accounting, monitoring, and configuration
- A Java™ GUI (created using these same APIs)
- Policy support following DEN² schema (LDAP enabled)

2. Directory-Enabled Networking (DEN) initiative details can be found at http://www.dmtf.org/standards/standard_den.php

Figure 4-1: Solaris Bandwidth Manager module

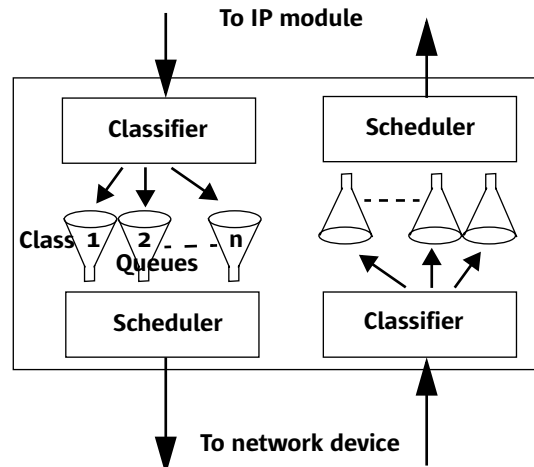


Solaris Bandwidth Manager was designed and implemented as a separate STREAMS module that is inserted into the TCP/IP stack when Solaris Bandwidth Manager is installed and configured (see figure 1).

This STREAMS module's two main functional components are the classifier and the scheduler:

- The Classifier separates IP packets into different classes, based on various selectors such as source or destination IP address, port number, or URL. Each class corresponds to a software queue in which the packets can be temporarily stored (if they can't be sent immediately due to congestion or policy).
- The Scheduler is responsible for dequeuing packets and sending them according to predefined policies. Solaris Bandwidth Manager uses a Class-Based Queueing (CBC) algorithm, which allows hierarchical policies to be defined with guaranteed bandwidth allocation, traffic shaping, the borrowing of unused bandwidth from other classes, hierarchical bandwidth allocation, and bandwidth allocations configured as link percentages or absolute bit rates.

Figure 4-2: Solaris Bandwidth Manager Functional Diagram



Appendix A

Feature Comparison Between Solaris 9 IPQoS and Solaris Bandwidth Manager

	Solaris 9 IPQoS	Solaris Bandwidth Manager
Design	Fully integrated	Layered
Accounting and monitoring	exactt, kstat	Netflow, SNMP, APIs
uid/projectid classification	Yes	No
IPSec: classify before encryption	Yes	No
802.1D priority marking	Yes ^A	No
Traffic policing/shaping	Policing only	Yes
Policy schema (DEN) with LDAP support	No	Yes
URL based classification	No	Yes
Administration	cmd line	cmd line, GUI, APIs

	Solaris 9 IPQoS	Solaris Bandwidth Manager
Transparent mode	No	Yes
Product availability	Free, integrated in Solaris	Available at extra cost
Solaris releases supported	Solaris 9 and beyond	Solaris 2.5 through Solaris 8

A. On supported NICs only (must use VLAN ethernet headers)

Appendix B

IP Type of Service (ToS) Application APIs

It is possible for applications running on the Solaris Operating System to directly influence its desired forwarding behavior by setting the IP type of service byte directly from within the application. Solaris applications using the socket library (libsocket), as well as Java applications (as of the Java 2 Platform, Standard Edition (J2SE™) 1.4.0 release), support an interface for this.

Solaris Socket Library

Application programmers can set the ToS field in the IP header of outgoing datagrams using the `IP_TOS` option of the IP module (see `IP(7s)`), settable using the `setsockopt(3SOCKET)` library call.

```
#include <sys/socket.h>
#include <netinet/ip.h>

s = socket(PF_INET, SOCK_STREAM, proto);

int tos = IPTOS_LOWDELAY; /* see <netinet/ip.h> */
setsockopt(s, IPPROTO_IP, IP_TOS, &tos, sizeof(tos));
```

The `getsockopt(3SOCKET)` call will return the last value that was set using the `setsockopt` call, not the ToS value of incoming datagrams.

Java Technology

In Java technology, as of the J2SE 1.4.0 release, programmers can use the `setTrafficClass` method to set the ToS of outgoing datagrams.

```
import java.net.*;

int IPTOS_LOWDELAY = 0x10;

DatagramSocket s = new DatagramSocket();
s.setTrafficClass(IPTOS_LOWDELAY);
```

The `getTrafficClass` method returns the last ToS value that was set using the `setTrafficClass` method.

The `setsockopt` and `setTrafficClass` calls are not DiffServ aware, and there is no feedback mechanism back to the application about operational behavior such as congestion. Solaris 9 IPQoS or the Solaris Bandwidth Manager software can be utilized to classify using the ToS set by an application, which may also rewrite the DSCP/ToS value if necessary. This can help prevent resource intensive applications from dominating the available network resources.

SUN™ Copyright 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Java, J2SE, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

SUN™ Copyright 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, Californie 95054 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, Java, J2SE, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



Please
Recycle



Adobe PostScript

Learn More

Get the inside story on the trends and technologies shaping the future of computing by signing up for the Sun Inner Circle program. You'll receive a monthly newsletter packed with information on the latest innovations, plus access to a wealth of resources. Register today to join the Sun Inner Circle Program at sun.com/joinic.

To receive additional information on Sun software, products, programs, and solutions, visit sun.com/software.

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA Phone 800 786-7638 or +1 512 434-1577 Web sun.com



Sun Worldwide Sales Offices: Africa (North, West and Central) +33-13-067-4680, Argentina +5411-4317-5600, Australia +61-2-9844-5000, Austria +43-1-60563-0, Belgium +32-2-704-8000, Brazil +55-11-5187-2100, Canada +905-477-6745, Chile +56-2-3724500, Colombia +571-629-2323, Commonwealth of Independent States +7-502-935-8411, Czech Republic +420-2-3300-9311, Denmark +45 4556 5000, Egypt +202-570-9442, Estonia +372-6-308-900, Finland +358-9-525-561, France +33-134-03-00-00, Germany +49-89-46008-0, Greece +30-1-618-8111, Hungary +36-1-489-8900, Iceland +354-563-3010, India-Bangalore +91-80-2298989/2295454; New Delhi +91-11-6106000; Mumbai +91-22-697-8111, Ireland +353-1-8055-666, Israel +972-9-9710500, Italy +39-02-641511, Japan +81-3-5717-5000, Kazakhstan +7-3272-466774, Korea +822-2193-5114, Latvia +371-750-3700, Lithuania +370-729-8468, Luxembourg +352-49 11 33 1, Malaysia +603-21161888, Mexico +52-5-258-6100, The Netherlands +00-31-33-45-15-000, New Zealand-Auckland +64-9-976-6800; Wellington +64-4-462-0780, Norway +47 23 36 96 00, People's Republic of China-Beijing +86-10-6803-5588; Chengdu +86-28-619-9333; Guangzhou +86-20-8755-5900; Shanghai +86-21-6466-1228; Hong Kong +852-2202-6688, Poland +48-22-8747800, Portugal +351-21-4134000, Russia +7-502-935-8411, Singapore +65-6438-1888, Slovak Republic +421-2-4342-9485, South Africa +27 11 256-6300, Spain +34-91-596-9900, Sweden +46-8-631-10-00, Switzerland-German 41-1-908-90-00; French 41-22-999-0444, Taiwan +886-2-8732-9933, Thailand +662-344-6888, Turkey +90-212-335-22-00, United Arab Emirates +9714-3366333, United Kingdom +44 0 1252 420000, United States +1-800-555-9SUN or +1-650-960-1300, Venezuela +58-2-905-3800