ポリシーサーバ PolicyXpert における Diffserv ポリシーとそのくみあわせ

金田　泰[†]，ブライアン・オキーフ[‡]

[†]日立製作所 研究開発本部 IP ネットワーク研究センタ
〒244-0817 横浜市戸塚区吉田町 292 番地

[‡]Hewlett-Packard Company
3404 East Harmony Road MS A2 Fort Collins, CO 80528-9599, USA

E-mail: [†]kanada@crl.hitachi.co.jp, [‡]bokeefe@cnd.hp.com

あらまし ネットワークのポリシー制御において，Diffserv のようなプログラムしカスタマイズすることが可能なネットワーク機能を表現するために，複数のポリシーをくみあわせる必要が生じることがある．PolicyXpert というポリシーサーバにおいて，3 種類のポリシーと，ポリシーを接続するための 3 種類の仮想フローラベル (VFL) とを設計・実装した．ポリシーのくみあわせによって複雑な Diffserv ポリシーが表現可能になる．また，ポリシーをくみあわせることにより，DSCP にもとづくサービスクラスのサブクラスの定義が可能になり，サービスポリシーと加入者ポリシーとの分離が可能になっている．しかも，Diffserv ポリシーを慎重に設計したため，単純な Diffserv ポリシーは単純なかたちであらわせるようになった．

キーワード ポリシーベース・ネットワーキング，ポリシーサーバ，Diffserv，ポリシー結合，仮想フローラベル．

Diffserv Policies and Their Combinations in a Policy Server Called PolixyXpert

Yasusi Kanada[†], Brian J. O'Keefe[‡]

[†]IP Network Research Center, Research & Development Group, Hitachi, Ltd.
Totsuka-ku Yoshida-cho 292, Yokohama 244-0817, Japan

[‡]Hewlett-Packard Company
3404 East Harmony Road MS A2 Fort Collins, CO 80528-9599, USA

E-mail: [†]kanada@crl.hitachi.co.jp, [‡]bokeefe@cnd.hp.com

Abstract In policy-based networking, policies sometimes have to be combined and applied in cooperation to represent such programmable and customizable network functions as Diffserv. For a policy server called PolicyXpert, we have designed and implemented three types of policies and three types of virtual flow labels (VFLs) to connect the policy rules. The policy combination enables the representation of complex Diffserv policies. Policy combination also allows sub-classing of DSCP-based service classes, and the separation of service and subscriber policies. The careful design of Diffserv policies has enabled simple Diffserv policies to be represented in a simple form.

Key words Policy-based networking, Policy server, Differentiated Services (Diffserv), Policy combination, Virtual flow label.

1. INTRODUCTION

Policy-based network management (PBNM) is a promising technology because it provides the following benefits. Firstly, it is easy to create, to modify, and to delete policies dynamically without interfering other policies because they are *rule-based*. A rule is a fine-grained module that can be added, deleted, or modified independent of other rules. Secondly, the amount of network configuration task can be reduced by using policies because one policy can be used for policy targets that are of various types, i.e., network nodes or interfaces, and that have been developed by a variety of vendors. Thirdly, heterogeneous networks can be managed according to a unified set of policies that follows the IETF (Internet Engineering Task Force) standards. Policies are modeled by the Policy Framework Working Group of the IETF [Moo 01, Sni 00]. A protocol called COPS (Common Open Policy Services) [Dur 00], its usage called COPS-RSVP [Her 00] and COPS-PR [Cha 01], and data formats conveyed by COPS-PR, which are called PIBs (policy information bases), are also standardized by the IETF.

In programmable and customizable networks, two or more policies often work in cooperation. For example, in a QoS-assured network service such as Diffserv (Differentiated Services) [Car 98, Ber 99], packet flows from service subscribers are classified and policed (i.e., limited to a certain bandwidth) at an edge router, and queued and scheduled in each router that the flow passes through. Thus, policies for classification, policing, and queuing/scheduling must cooperate to assure QoS. If the service is typical Diffserv, the policy for classification specifies the class or the DSCP (Diffserv Code Points) [Nic 98] of the flow, and the policy for queuing/scheduling specifies the testing of the DSCPs to determine the algorithms and parameters for queuing and scheduling required by packets in that class. These policies can be regarded as components of a network-wide QoS policy. Although DSCPs can be used for implicit cooperation of policies, other services may require explicit cooperation.

Lupu and Sloman [Lup 99] developed methods for handling policy conflicts. Conflicts are types of relationships between policies. They are usually negative relationships because they cause inconsistencies among policies and because they accidentally and implicitly occur. In contrast, the concept of policy combination [Kan 01a] was developed for explicitly specifying a *positive* relationship between policies [Kan 00a, Kan 00b]. Conventionally, no methods for combining policies explicitly have been developed.

Policies can be complex as explained above. If they are complex, they should be built from building-block policies. The structure of the combined policy may not be very simple. However, a simple policy should be simple; i.e., if the nature of a policy is simple, it should be represented in a simple form.

This paper explains the three types of provisioning Diffserv policies and the methods and application cases of policy combination in a policy server (PDP) called OpenView PolicyXpert and JP1/PolicyXpert[1]. The goal of the policy design in this policy server is to enable both policy combination in complex policies and simple representation in simple policies using a concise set of policies. Policy and system architectures of PolicyXpert overviewed in Section 2, and the three types of Diffserv policies are explained in Section 3. The method of policy combination in PolicyXpert is explained in Section 4, and it application cases are explained in Section 5.

2. POLICY AND SYSTEM ARCHITECTURES

A policy is a sequence of condition-action rules in PolicyXpert. An example of a condition-action rule is as follows:

```
if (Source_IP_address is 192.168.1.1) {      -- condition
    DSCP = 10;                               -- action
}.
```

This rule marks DSCP 10 on (the packets in) a flow from IP address 192.168.1.1.

A policy **P** may be represented as:

P = {rule1, rule2, … }.

The conditions for applying rules 1, 2, … are evaluated from top to bottom, and only the action which corresponds to the first condition to be matched is taken. Policies in this system mostly conforms with the IETF Policy Core Information Model (PCIM) [Moo 01].

The outline of PolicyXpert architecture is illustrated in **Figure 1**. Policies are defined by the administrator or operators using the console. They are inputted to the server (PDP) and stored into the policy database. They are deployed to proxy or embedded policy agents (PEP) and the network devices are configured by the agents. The server also manages network interfaces of the devices by using the information managed and sent by the agents.

3. THREE TYPES OF DIFFSERV POLICIES

There are three policy types for Diffserv in PolicyXpert 2.0 [HP 00] and later versions. See **Figure 2**.

1. *Traffic Classifier (CL) Policy*
   A CL Policy classifies (the packets in) packet flows and assigns labels called CIDs (Classifier Identifiers. See Section 4.1). A CL Policy is usually deployed to edge or border interfaces (i.e., interfaces that are connected to points outside the Diffserv domain) and applies to inbound traffic.

[1] OpenView and PolicyXpert are trademarks of Hewlett-Packard Company. JP1 is a trademark of Hitachi, Ltd. PolicyXpert Version 2.0 was developed jointly by Hewlett-Packard and Hitachi.

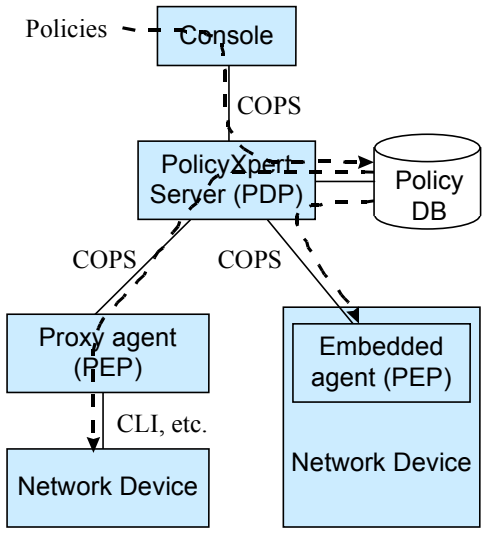


Figure 1: Architecture of PolicyXpert

2. *Traffic Conditioner (TC) Policy*
   A TC Policy meters, marks, and/or drops packets absolutely (i.e., unconditionally). TC Policies, too, are usually deployed to edge or border interfaces and apply to inbound traffic.
3. *Queue Control (QC) Policy*
   A QC Policy queues and schedules, or drops packets randomly (by using the WRED or a similar algorithm). A QC Policy is usually deployed to core interfaces (i.e., interfaces that are connected to other interfaces within the Diffserv domain) and applies to outbound traffic. A QC Policy rule can be regarded as (a model of) a queue or scheduler; i.e., a traffic control object.

TC and QC Policies are natural representation of Diffserv functions as policies; i.e., collections of condition-action rules. So the Diffserv policies mostly conform with the IETF Diffserv models, i.e., the conceptual router model [Ber 01], Diffserv MIB [Bak 01] and PIB [Fin 01], and QoS Policy Information Model (QPIM) [Sni 00]. In this policy server, only one instance of a policy type can be deployed at one network interface.

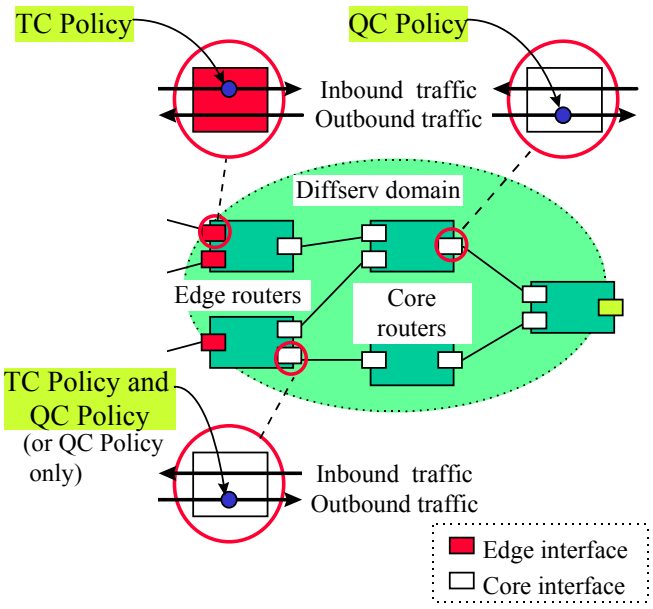


Figure 2: Diffserv policy types and their deployment

Examples of these policies are given below. Although a GUI is used for creating and editing policies in PolicyXpert, the policies are described in a C-like language here for concise description.[1]

1. *Example of a CL Policy rule*
   The following rule (1.1) assigns the CID value "EF_CID" to flows come from IP address 192.168.1.1.

```
if (Source_IP_address is 192.168.1.1) {
    CID = "EF_CID";
}.                                                    (1.1)
```

   Marking a CID is the only function of the CL Policy, and it is usually used as a component of a larger policy.

2. *Examples of TC Policy rules*

- *A simple (stand-alone) rule*

The following rule (2.1) is a simple TC Policy rule and marks DSCP 10 on packets.

```
if (Source_IP_address is 192.168.1.1) { DSCP = 10; }.
                                                      (2.1)
```

This rule can be used as a stand-alone rule; i.e., in a given device, no other rule may be applied in cooperation with this rule.

- *A more complex rule*

Rule (2.2), shown below, is applied to flows to which the CID value "EF_CID" has been attached to.

```
if (CID is "EF_CID") {
    if (InformationRate <= 10 Mbps) {
        DSCP = "EF";    -- marking
    } else {
        Discard;        -- absolute drop
    };
}.                                                    (2.2)
```

This rule may thus be combined with rule (2.1). Rule (2.2) meters the traffic and marks the DSCP "EF" on the packets in the first 10 Mbps of the traffic, and discards other packets.[2] This rule can be used for an EF (Expedited Forwarding) service [Jac 99] of Diffserv, and it must be combined with a TC Policy rule such as rule (1.1).

---

[1] However, no such language is currently supported.

[2] Under PolicyXpert, if two or more CL Policy rules specify the CID value "EF_CID", the information rate of either of these flows can reach 10 Mbps, and the sum of these flows may exceed 10 Mbps. This is because rule (2.2) is conceptually copied before combination with other CL Policy rules, so that the original rule (2.2) works as a template. A more detailed description of the semantics is given in the PolicyXpert Users Guide [HP 00].

3. Examples of QC Policy rules

- *A simple (stand-alone) rule*

The following rule (3.1) applies a bounded priority queuing algorithm to the queuing and scheduling of packets with a DSCP of "EF".[1]

```
if (DSCP is "EF") {
    SchedulingAlgorithm = "B-PQ";
                -- bounded priority queuing
    Priority = 6;  -- means "high"
    ShapingRate = 20 Mbps;
}.                                          (3.1)
```

The traffic is then shaped to 20 Mbps. Rule (3.1) represent a queue that is connected to a priority scheduler that is not given as a rule.

- *A more complex rule*

Rule (3.2), shown below, represents a scheduling queue, and this rule specifies three discard levels; newly coming packets with a DSCP of "AF11" are discarded only when the queue is filled with 200 packets (100%), while newly coming packets with a DSCP of "AF12" are discarded when the queue contains 140 (70%) or more packets and newly coming packets with a DSCP of "AF13" are discarded when the queue contains 100 (50%) or more packets.

```
if (DSCP is ["AF11", "AF12", "AF13"]) {
    SchedulingAlgorithm = "A-BW";
    Max_Queue_Size = 200 packets;
    CommittedRate = 64 kbps;
            -- assured minimum rate
    DiscardAlgorithm = "Deterministic Discard";
    if (DSCP is "AF11") {
        DiscardLevel = 100%;
            -- allowed to use whole queue
    } elsif (DSCP is "AF12") {
        DiscardLevel = 70%;
            -- allowed to use 70% of the queue
    } elsif (DSCP is "AF13") {
        DiscardLevel = 50%;  };
            -- allowed to use 50% of the queue
}.
```

"Deterministic Discard" is specified as the discard algorithm. This specifies a non-random algorithm for dropping packets. This rule can be used for an AF (Assured Forwarding) service [Hei 99] of Diffserv. A random discard method such as the weighted random early discard (WRED) can be specified instead of deterministic discard too. The reason that the all the discard levels are specified in a rule is that rule (3.2) also represents a queue; i.e., the discard levels are specified for a single queue. If they are separated into different rules, they specify different queues.

Example of a rule that represent a scheduler will be shown in Section 5.2.

[1] The actual value of DSCP is a 6-bit number. However, a name can be given by using a parameter group in PolicyXpert.

4. METHOD OF POLICY COMBINATION

To combine policies, both the dataflow and the control flow between policies must be specified.

4.1 Specification of dataflow between combined policies

A specific dataflow of packets can be detected by using *flow labels* [Kan 00b]. Flow labels are labels attached to a packet or flow. They are used for selecting a rule from a policy. Flow labels are of two types (illustrated in **Figure 3**).

1) *Real flow labels*: labels written inside the packet. A DSCP is an example of a real flow label.

2) *Virtual flow labels* (*VFLs*): labels external to the packet.

A real flow label is conveyed by packets, so the two policies to cooperate can exist in different network nodes. However, a VFL is not conveyed by packets themselves, so the policies to cooperate must exist in the same network node unless the virtual tag value is conveyed by some other means, such as wavelength, physical location, and so on. We focus on the usage of VFLs below.

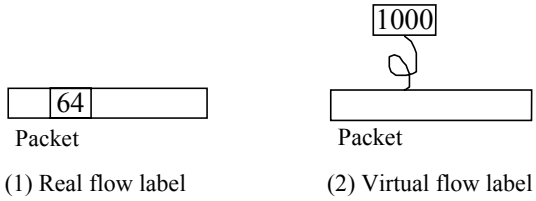


Figure 3: Two types of flow labels

Dataflows between policies are specified by defining and using VFL values in the rules of the policies. If Policy 2 is executed after Policy 1 (illustrated in **Figure 4**), a VFL can be defined by one or more of the rules (in actions) of Policy 1, and the VFL can be referred to in one or more of the rules (in conditions) of Policy 2.[2]

VFLs in PolicyXpert are classified into three categories:

1. *Classifier ID* (*CID*): A CID combines rules in CL and TC policies. For example, rules (1.1) and (1.2) in Section 3 are combined by the CID "EF_CID".
2. *Traffic ID* (*TID*): A TID combines rules in a TC Policy.
3. *Queue Set ID* (*QID*): A QID combines rules in a QC Policy. An example of this combination is shown in Section 5.2.

Policy combinations created by these VFLs are illustrated

[2] Two rules in Policy 2 should not be combined to rules in Policy 1 in reversed order. Otherwise, difficult semantic problems may occur. This means, when there are rules 1a and 1b in Policy 1 and rules 2a and 2b in Policy 2, and rule 1a precedes 1b, rule 2a precedes rule 2b, rule 1a assigns VFL a, rule 2b assigns VFL b, then if rule 2a refers to VFL b, rule 2b should not refer to VFL a because the reference order is the reversed assignment order. If rule 2a refers to VFL a, rule 2b can refer to VFL b because the orders coincide.

in **Figure 5**. A character string is used for VFL values in PolicyXpert. The string values are usually translated into other type of data, such as integral values by a policy agent (PEP). TIDs and QIDs only combine rules within a single TC or QC Policy because there is only one instance of a TC or QC Policy for an interface.

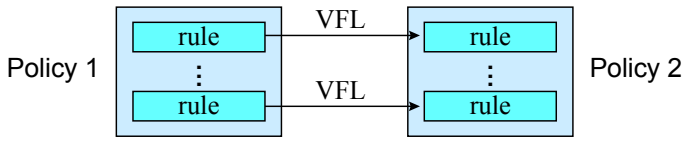


Figure 4: Connection of policies using VFLs

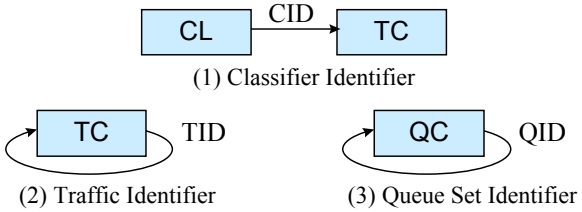


Figure 5: Three types of VFLs

4.2 Specification of control flow between combined policies

Flow of control can be explicitly specified by using a properly-defined policy language [Kan 00b]. For example, if CL1, TC1, and QC1 are policies, then the following declaration specifies a flow of control:

```
order CL1 -> TC1 -> QC1.
```

However, the order of policy evaluation can be predefined as part of the definition of a specific policy. The order of CL, TC and QC Policies is predefined as shown in **Figure 6**; i.e., a CL Policy can be followed by a TC Policy, a TC Policy can be followed by a TC or QC Policy, and a QC Policy can be followed by a QC Policy or no policy.

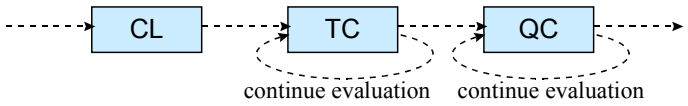


Figure 6: Control flow between Diffserv policies

There is nondeterminacy (i.e., there are alternatives) in the execution order of TC and QC Policies. To resolve this nondeterminacy, "continue evaluation" must be explicitly specified when a policy evaluation is repeated. For example, the following rule in a QC Policy is combined with another rule (that tests the QID "shape2") in the QC Policy.

```
if (DSCP is "AF11" or "AF12" or "AF13") {
    SchedulingAlgorithm = "A-BW";
    MaxQueueSize = 200 packets;
    CommittedRate = 64 kbps;
    QID = "shape2";
    Continue evaluation;
}.
```

If no "continue evaluation" is specified, the rule is not followed by another QC Policy rule.

5. APPLICATION OF POLICY COMBINATION

Two cases of the application of policy combination in PolicyXpert are explained here.

5.1 Separation of subscriber and service policies

Both network services and service subscribers, i.e., end customers, can be managed by using policies. Policies for service subscribers can be separated from the service policies by using CL and TC Policies as well as CIDs (shown in **Figure 7**).

In a Diffserv network, three service classes, i.e., gold, silver, and best-effort classes, can be defined. The same DSCP can be used for both gold and silver classes, but the policing rates for them, which are specified by TC Policy rules, can be different; e.g., a gold traffic is policed to 1 Mbps, but a silver traffic is policed to 128 kbps. Then, two different DSCPs are used, and three different CIDs, "G", "S", and "B" (which represent subclasses of DSCP-based classes), are used for gold, silver, and best-effort classes. Service properties can be defined by the network administrator in a service policy, which is implemented by using a TC Policy; each class of services is specified by a TC Policy rule. Subscriber properties can be defined by the network operators in subscriber policies, which are implemented by using CL Policies; each subscriber is specified by a CL Policy rule. CIDs are used for mapping or aggregating subscribers into service classes. The TC Policy can then be deployed to all inbound edge interfaces of the Diffserv network. Each CL Policy can be deployed to an edge interface and contain rules connected to the service policy rules in the TC Policy by CIDs. In Figure 7, CL Policies 1, 2, and 3 (subscriber policies) are defined, and they are deployed to three edge interfaces. There is only one TC Policy (service policy), and it is deployed to the same interfaces as the CL Policies.

When a subscriber is added or removed, the network operator can modify only the relevant CL Policy and need not modify the service policy. Particularly, multiple service classes that share a DSCP are separated by using CIDs. This separation of subscriber and service policies clearly separates the task of the network administrator from the task of the network operator. Subscriber and service policies are separated by using VFLs, but the policies cooperates following a uniform policy semantics.

5.2 Hierarchical shapers and policers

In multi-service networks, hierarchical schedulers and shapers can be used for harmonizing various types of traffic. These functions can be represented by using QIDs and a QC Policy. Each QC Policy rule represents a simple queuing or scheduling method. QC Policy rules can be combined by QIDs to represent a complex queu-

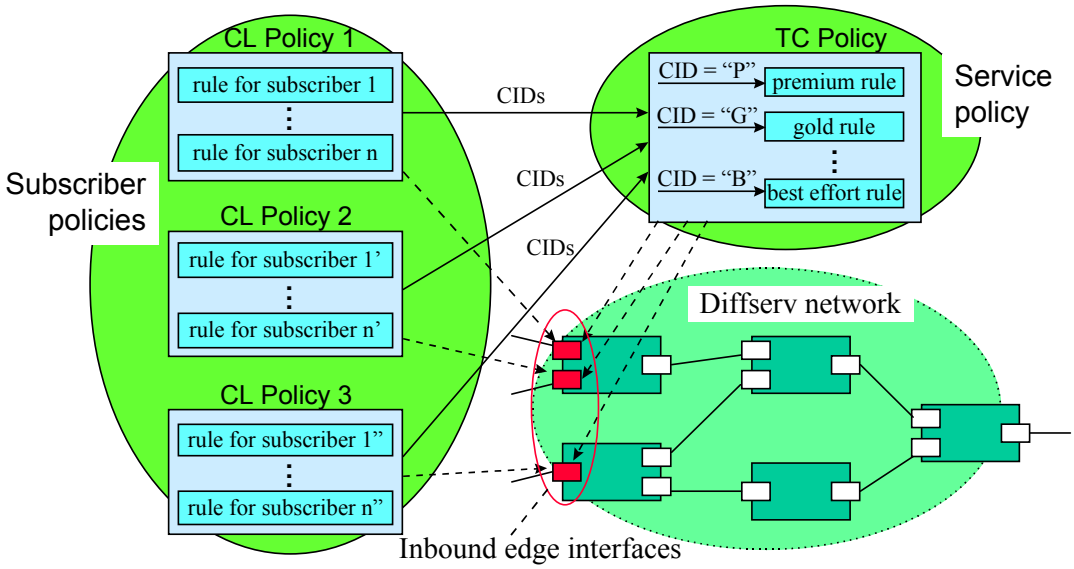
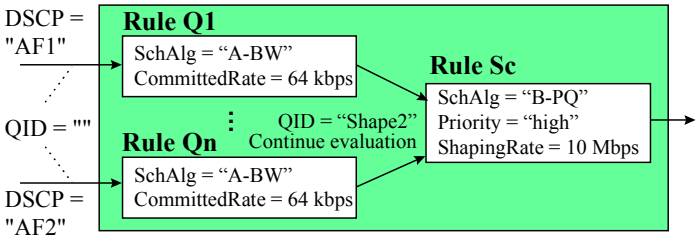


Figure 7: Separation of subscriber and service policies

ing/scheduling method.

For example, a hierarchical shaper can be outlined as shown in **Figure 8**. This QC Policy consists of n + 1 rules. Rules Q1, …, Qn receive input traffic, and output traffic shaped at a maximum of 64 kbps by using a bandwidth fair queuing (A-BW[1]) method. Here, the input traffic is assumed to have the QID value "" (empty string), the output traffic has the QID value "Shape2", and "continue evaluation" is specified in each of rules Q1, …, Qn. A-BW can be mapped to an appropriate scheduling (queuing) method implemented at the given network node. Rule Sc inputs the aggregation of the shaped traffic from Q1, …, Qn,[2] and outputs traffic at a maximum of 10 Mbps by using a bounded priority queuing (B-PQ) method.




```
S = { Q1:  if (QID is "" && DSCP is "AF1") {
             Scheduling_Algorithm = "A-BW";
             Committed_Rate = 64 kbps;
             QID = "Shape2";
             Enqueue;
           },
           …,
      Qn:  if (QID is "" && DSCP is "AF2") {
             Scheduling_Algorithm = "A-BW";
             Committed _Rate = 64 kbps;
             QID = "Shape2";
             Enqueue;
           },
      Sc:  if (QID is "Shape2") {
             Scheduling_Algorithm = "B-PQ";
             Priority = "high";
             Shaping_Rate = 10 Mbps;
             QID = "Outgoing";
             Enqueue;
           } }
```

Figure 8: A hierarchical shaper

Each of rules Q1, …, Qn models a queue, and rule Sc models an A-BW scheduler that is followed by a B-PQ scheduler that is not given explicitly. Other scheduling methods, i.e., strict priority queuing (S-PQ) and per-flow bandwidth fair queuing (P-BW), can also be specified in a QC Policy rule.

A hierarchical policer can be represented in a similar way to the above shaper, but the details are omitted here.

Note that, although a hierarchical shaping, scheduling, or policing policy is complex, a simpler function, such as a non-hierarchical shaping, scheduling, or policing function, or marking function, can be represented by only a single rule.

[1] A-BW is an abbreviation of "aggregated bandwidth fair queuing". "Aggregated" means that this scheduling algorithm distinguishes aggregated flows (by using DSCPs) but does not distinguish microflows. This is different from the other bandwidth fair queuing algorithm called "per-flow bandwidth fair queuing" (P-BW), which distinguishes microflows and can be used for Packeteer's PacketShaper™.

[2] Rule Sc is shared among rules Q1, …, Qn, and is not copied; i.e., all the queues that are represented by Q1, …, Qn are connected to the same scheduler that is represented by Sc. The semantics of a QID differ in this way from those of a CID as described in Section 3.

6. CONCLUSION

Policy combination is required to represent programmable and customizable network functions such as those provided by Diffserv. In PolicyXpert, policies of three types (i.e., CL, TC, and QC) and VFLs of three types (i.e., CIDs, TIDs, and QIDs), for connecting policy rules, are defined for Diffserv. The policy combination enables the representation of complex Diffserv policies. TC and QC Policies, and TIDs and QIDs can be used in constructing such representations. Policy combination in PolicyXpert also allows sub-classing of DSCP-based service classes and the separation of service and subscriber policies. CL and TC Policies, and CIDs are available for this purpose. The careful design of Diffserv policies has enabled simple Diffserv policies to be represented in a simple form.

Future work on PolicyXpert will include the refinement of the semantics of policy combination, especially the evaluation order of rules that refers to VFLs.

ACKNOWLEDGMENT

We thank Toshio Shimojou of the Enterprise Server Division, Hitachi, Ltd., and Rick Roeling of the Hewlett-Packard Company for discussing on the policy design of PolicyXpert with us.

REFERENCES

[Bak 01] Baker, F., Chan, K., and Smith, A., "Management Information Base for the Differentiated Services Architecture", draft-ietf-diffserv-mib-09.txt, *Internet Draft*, IETF, March 2001.

[Ber 99] Bernet, Y., Binder, J., Blake, S., Carlson, M., Carpenter, B. E., Keshav, S., Ohlman, B, Verma, D., Wang, Z., and Weiss, W., "A Framework for Differentiated Services", draft-ietf-diffserv-framework-02.txt, *Internet Draft*, IETF, February 1999.

[Ber 01] Bernet, Y., Blake, S., Grossman, D., and Smith, A., "An Informal Management Model for Diffserv Routers", draft-ietf-diffserv-model-06.txt, *Internet Draft*, IETF, February 2001.

[Car 98] Carlson, M., Weiss, W., Blake, S., Wang, Z., Black, D., and Davies, E., "An Architecture for Differentiated Services", RFC 2475, IETF, December 1998.

[Cha 01] Chan, K. H., Durham, D., Gai, S., Herzog, S., McCloghrie, K., Reichmeyer, F., Seligson, J., Smith, A., and Yavatkar, R., "COPS Usage for Policy Provisioning (COPS-PR)", RFC 3084, IETF, March 2001.

[Dur 00] Durham, D. (ed.), Boyle, J., Cohen, R., Herzog, S., Rajan, R., and Sastry, A., "The COPS (Common Open Policy Service) Protocol", RFC 2741, IETF, January 2000.

[Fin 01] Fine, M., McCloghrie, K., Seligson, J., Chan, K., Hahn, S., Smith, A., and Reichmeyer, F., "Differentiated Services Quality of Service Policy Information Base", draft-ietf-diffserv-pib-03.txt, *Internet Draft*, IETF, March 2000.

[Hei 99] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.

[Her 00] Herzog, S ., ed., Boyle, J., Cohen, R., Durham, D., Rajan, R., and Sastry, A., "COPS usage for RSVP", RFC 2749, IETF, January 2000.

[HP 00] HP OpenView PolicyXpert 2.0 Users Guide, Edition 1, Hewlett-Packard, October 2000.

[Jac 99] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB", RFC 2598, June 1999.

[Kan 00a] Kanada, Y., "A Representation of Network Node QoS Control Policies Using Rule-based Building Blocks", *International Workshop on Quality of Service 2000* (*IWQoS 2000*), pp. 161–163, June 2000.

[Kan 00b] Kanada, Y., "Two Rule-based Building-block Architectures for Policy-based Network Control", *2nd International Working Conference on Active Networks* (*IWAN 2000*), Lecture Notes in Computer Science, No. 1942, pp. 195–210, Springer, October 2000.

[Kan 01a] Kanada, Y., "Taxonomy and Description of Policy Combination Methods", *Workshop on Policies for Distributed Systems and Networks* (*Policy 2001*), Lecture Notes in Computer Science, No. 1995, pp. 171–184, Springer, January 2001.

[Kan 01b] Kanada, Y., and O'Keefe, B. J., "Combination of Diffserv Policies in OpenView/JP1 PolicyXpert", *5th Asia-Pacific Network Operations and Management Symposium* (*APNOMS 2001*).

[Lup 99] Lupu, E., and Sloman, M., "Conflicts in Policy-based Distributed Systems", *IEEE Trans. On Software Engineering*, Vol. 25, No. 6, pp. 852–869, 1999.

[Moo 01]Moore, B., Ellesson, E., Strassner, J., and Westerinen, A., "Policy Framework Core Information Model — Version 1 Specification", RFC 3060, IETF, February 2001.

[Nic 98] Nichols, K., Blake, S., Baker, F., and Black, D., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, IETF, December 1998.

[Sni 00] Snir, Y., Ramberg, Y., Strassner, J., and Cohen, R., "Policy Framework QoS Information Model", draft-ietf-policy-qos-info-model-02.txt, *Internet Draft*, IETF, November 2000.