ネットワーキングアーキテクチャワークショップ資料

Rule-based Modular Representation of QoS Policies*

Yasusi Kanada

Hitachi Ltd., Central Research Laboratory
1-280 Higashi-Koigakubo, Kokubunji, Tokyo 185-8601 Japan
Phone: +81-42-323-1111
E-mail: kanada@crl.hitachi.co.jp

Key words: QoS guarantee, differentiated services (DiffServ), policy control, rule-based language.

Abstract: To realize internet-protocol-based QoS-assured networks, using differentiated services under policy-based networking is a promising approach. A QoS policy server must work in multi-vendor environment. To use standard protocol, such as COPS or SNMP, between the policy server and routers is not sufficient, but also to define and to standardize high-level syntax and semantics, i.e., a language, is required for interoperability. This paper describes the outline of a rule-based language for this purpose. Policy rules can be defined in the policy server and can be deployed to routers or router proxies using this language through an appropriate protocol such as COPS, SNMP, or IIOP. The language consists of several types of rules, i.e., matching, policing (or metering), marking, discarding, and scheduling types, and linkage labels that connects rules. A MIB and/or PIB that simulates the language is also explained in this paper. The language will be implemented in near future.

1. Introduction

The Internet will be used much more often for mission-critical communications, and multimedia services will be much more important in the Internet in near future. Thus, guarantee or assurance of QoS (quality of service) will be essential for Internet-based services for commercial service providers and enterprises.

In the IETF (Internet Engineering Task Force), many working groups concern Internet QoS. Especially, the Integrated Services (IntServ) WG is working on per-flow QoS guarantee [Wro 97][She 97], and the differentiated services (DiffServ) WG is working on class-based QoS assurance [Ber 99a]. “Per-flow” means that each flow of packets between a source end-point, and a destination end-point is treated individually. “Class-based” means that flows are classified into service classes, and the flows in a same service class are treated in the same way. Per-flow control makes more accurate QoS control possible, but it requires much more resources in network nodes, such as routers. Because the resources are limited, class-based approach, i.e., DiffServ, seems to be more practical on the Internet.

To control QoS conditions in a network domain in which DiffServ take place, a policy server is used. The policy server configures routers in the domain by deploying policy rules that classify packets and assign QoS conditions to packets. There are several alternatives for the interface between a policy server and routers. SNMP [Cas 99][Har 99] or COPS [Boy 99] can be used as the protocol between them. APIs (application programming interfaces) can be specified for both policy-server side and router side. If CORBA is used for defining the APIs, IIOP [OMG 98] may be used for the protocol.

However, none of these alternatives themselves define the high-level syntax nor semantics of the policy rules, and the high-layer communication between a policy server and routers. The syntax and semantics of policy rules must be defined clearly, because a policy rule should be handled as whole and many grammatical constraints are embedded in a policy rule. Some constraints are required by the policy server or the operator, and others are required by routers. Because syntax and semantics are language issues, this means that a (programming) language must be defined. Because policies are represented by if-then rules, we think a rule-based language, such as used in developing expert systems, is appropriate for this purpose.

There are two design goals on defining rule-based language for DiffServ functions. The first goal is *interoperability*. A network domain may contain routers of many vendors, and the QoS functions of routers are mostly vendor-specific, even if they support standardized interface such as SNMP or COPS. Thus, a *vendor-independent* language should be defined and used in the interface for the sake of interoperability.

The second goal is *modularity*. There are two types of modularity: syntagmatic modularity and paradigmatic modularity. The words, syntagmatic and paradigmatic, are borrowed from Linguistics. *Syntagmatic modularity* means that a language expression, a policy rule in our case, can be divided into components and the components can be freely composed. The interface between a policy server and routers should be syntagmatically modular because the policy server should support most of functions of routers and so the interface should be *flexible*. If important but vendor-

* 規則にもとづくモジュラーな QoS ポリシーの表現，金田 泰 (日立製作所 中央研究所). キーワード: QoS 保証，DiffServ, ポリシー制御，ルールベース言語.

specific functions are not available through the interface, the policy server may be left unused or may be skipped because the required configuration can be done only by manual operation. *Paradigmatic modularity* means that an element of a set of syntactic entities that can be used in a specific part of the language expression can be added, removed, or modified without affecting other entities. Paradigmatic modularity is also important, but it is out of scope of this paper. So, it will not be explained more here.

To achieve interoperability and syntagmatic modularity, we have developed a representation of policy rules using *building block rules* and *linkage labels* (called virtual flow labels in [Kan 99]). A policy rule is a collection of building block rules connected by linkage labels. In this paper, a model of a DiffServ-ready network is described in Section 2, and a model of DiffServ-ready routers is described in Section 3, for the sake of clarifying the requirements. Then, outline of the rule-based language for DiffServ is proposed in Section 4. A MIB (management information base) and/or PIB (policy information base) that simulates the language is described in Section 5.

2. A Model of A DiffServ-ready Network

A DiffServ network domain can be modeled as in **Figure 1**. The domain is a part of networks in which the same set of PHBs (per-hop behaviors) [Bla 98] is used. A DSCP (differentiated services code point) [Nic 98] is assigned to each PHB in this domain. The network consists of routers. Because of simplicity, switches and hubs are ignored here. Computers are connected to routers. Some computers work as IP packet sources and others work as IP packet destinations. They are connected to the routers.

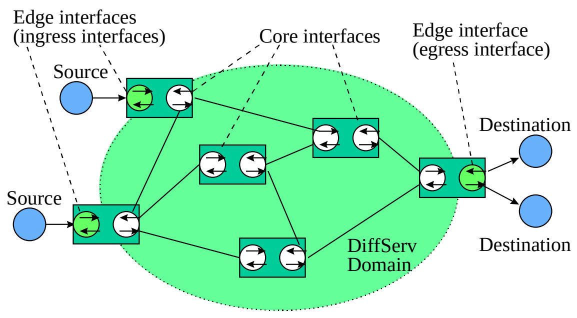

Figure 1. A network model

The (network) interfaces of the routers that are connected to computers are called *edge interfaces*, and the interfaces that are connected between routers are called *core interfaces*. Edge interfaces that are connected to packet sources are called *ingress interfaces*, and those connected to packet destinations are called *egress interfaces*. An interface may be used as both an ingress and egress interfaces, and it may be used as both edge and core interfaces. Routers that have edge interfaces are called *edge routers*, and routers that have only core interfaces are called *core routers*.

In this model, IP packets are classified at ingress interfaces, and are marked in their DS field (differentiated services field), which was formerly called ToS (type of services) field. The value in the DS field is called DSCP. The DSCP indicates the service class that the packet belongs to. At core interfaces, the QoS conditions of the packets are controlled according to the DSCP. The DS field may be cleared at egress interfaces.

IP packets are classified by using a *classifier*. A classifier uses a set of matching conditions, and each condition corresponds to a marking action, or a DSCP. Each pair of condition and action can be regarded as an if-then rule:

if (condition) action;

This rule works for each packet. The behavior of an interface can be specified using a set of if-then rules.

Classifiers used at ingress interfaces are called *MF classifiers* (multi-field classifiers). An MF classifier tests mainly the following five items: the source IP address, the destination IP address, the IP protocol, the source IP port, and the destination IP port.[1] The source and destination IP addresses may be a specific address, a subnet address, or an address range. The source and destination port may be a specific port number or a range of port numbers. The action taken as the result of MF classification is to assign a DSCP to a packet. The action may also include assigning a queue priority, packet discarding conditions, scheduling conditions, and so on, to the packet. An example rule is:

if (Source IP == 192.168.0.1 && Source port == 80)
DSCP = 46;

Classifiers are also used at core and egress interfaces. They are called *BA classifiers* (basic aggregate classifiers). A BA classifier tests the DSCP. The action taken as the result of BA classification is to assign a queue priority, and it may be to assign packet discarding conditions, scheduling or shaping conditions, and so on. An example rule is:

if (DSCP == 46) queue_priority = 6;

Rules with MF and BA classifiers are deployed by a policy server (**Figure 2**). A policy server deploys rules in accordance with network operator's com-

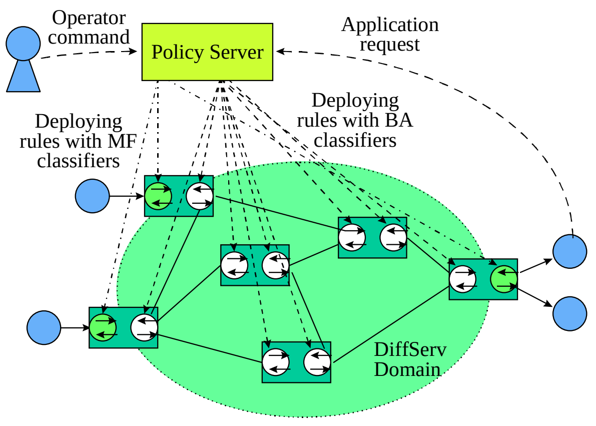

Figure 2. Centralized control over a DiffServ domain

[1] If the IP protocol is not TCP nor UDP, the source and destination ports are not supplied.

mands or requests of application programs.

3. A Model of DiffServ-ready Routers

A conceptual model of DiffServ-ready routers (**Figure 3**) [Kan 99] is explained here. The interface language must support the functions of this model. This model is similar to those described in the conceptual router model draft [Ber 99b] or in the DiffServ MIB draft [Bak 99], but there are some difference in details.

In this router model, a router consists of a switch fabric (and/or routing processor) and two or more network interfaces. The interfaces are connected to the switch fabric and one or more lines, and each interface consists of an inbound and outbound units.

Each inbound/outbound unit has three subunits: a control unit, a condition unit, and an action unit (**Figure 4**). The control unit contains lists of if-then rules (policy rules), which are programs to control the condition and action units. The policy-rule storage may be shared between inbound and outbound units, or by all the interfaces. All the policy-rule storage components in a router are assumed to have the same set of rules and a control unit uses a subset of them.

The condition unit contains matchers and meters. The classifier classifies packets into individual flows according to the matching rules supplied by the control unit. The meters work on a classified flow. A meter may count the number of packets or bytes in a queue, tests conformity with a leaky/token bucket condition, or tests other conditions on a stream. If the condition supplied by the meter meets (or does not meet), a linkage label that indicates the conformity (or non-conformity) is assigned to the packet.

The action unit, which can also be called the traffic-control block (TCB), consists of markers, discarders, and a scheduler that contains queues. A marker rewrites the DS field according to the condition outputted by the condition unit. A discarder discards packets when a metering condition is not satisfied or when the scheduler decides to discard packets. The scheduler buffers packets, and outputs them to the line. The scheduler discards a packet that overflows from the queue, or that the linkage label meets an early discard condition. For example, if the assured-forwarding per-hop behaviors (AF PHBs) [Hei 99] of the DiffServ model are implemented on the router, a queue must support multiple discard conditions. Early discard conditions enable this. Both deterministic discard and random early discard (RED) methods [Bra 99] are supported by this MIB.

4. A Rule-based Language for PS-to-Router Interface

An interface between a policy server and DiffServ-ready routers is defined as a rule-based language here. In this language, a policy rule to be deployed is described using building block rules and linkage labels that connects them.

4.1 Outline of the language

A policy rule may contain one or more meters, and may contain multiple actions depending on the result of metering. For example, the following rule contains a meter (“Average_rate <= 1Mbps”) and two actions (“DSCP = 46; …” and “discard_all”).

```
if (Source_ip == 192.168.1.*) {
    if (Average_rate <= 1Mbps) {
        DSCP = 46;        // EF
        queue_priority = 6;
    } else {
        discard_all;
    };
};
```

The above rule is already too complicated to be expressed as an atomic rule in the interface language. A more complicated rule may easily be constructed. Such rules must be expressed using smaller atomic units. If such rules were expressed using ready-made language constructs, they would be vendor-specific and non-interoperable.

There are two methods for expressing such a complicated rule by simpler language elements.

- *Recursion*
  If the action part of a rule may contain rules, complicated conditions and actions can be expressed. For example, in the above rule, both the outer and inner if-constructs are rules. The action part of the outer rule contains the inner rule.
- *Rule decomposition*

```
   +-----------------------------------------------------------+
   | Router                                                    |
   |                                                           |
   |  Logical interface    Switch fabric    Logical interface  |
   | +-----------------+(Routing processor)+-----------------+ |
   | |                 |     +-------+     |                 | |
   | | +-------------+ |     |\     /|     | +-------------+ | |
------>|Inbound unit |------>| \   / |<------|Inbound unit |<------
   | | +-------------+ |     |  \ /  |     | +-------------+ | |
   | |                 |     |   X   |     |                 | |
   | | +-------------+ |     |  / \  |     | +-------------+ | |
<------|Outbound unit|<------| /   \ |------>|Outbound unit|------>
   | | +-------------+ |     |/     \|     | +-------------+ | |
   | |                 |     +-------+     |                 | |
   | +-----------------+                   +-----------------+ |
   |                                                           |
   +-----------------------------------------------------------+
```

Figure 3. Conceptual model of a QoS-ready router

```
   +-----------------------------------------------------------+
   | Inbound/outbound unit                                     |
   |                                         +---------------+ |
   |                 +--------------------+  |  Policy rule  | |
   |                 |  Control unit      |--|    storage    | |
   |                 +--------------------+  |(if-then rules)| |
   |                   |                |    +---------------+ |
   |                   |                v                      |
   |                   |        +---------------------------+ |
   |                   v        | Action unit               | |
   | +-------------------------+ |   (Traffic control block) | |
   | | Condition unit          | |     +--------+            | |
   | |                         | | +-->|Droppers|<---+       | |
   | | +-----------+  +------+ | | |   +--------+    |       | |
------>|Classifier |->|Meters|-----+                 |       | |
   | | +-----------+  +------+ | | |  +-------+  +---------+ | |
   | |                         | | +->|Markers|->|Scheduler|------>
   | +-------------------------+ |    +-------+  +---------+ | |
   |                             |           (Queues)      | |
   |                             +---------------------------+ |
   |                                                           |
   +-----------------------------------------------------------+
```

Figure 4. The structure of an inbound/outbound unit of the router

A complicated rule may be decomposed into multiple simple rules.

We believe the latter is better because of two reasons:

1) A recursive rule is often difficult to be understood by operators (users).
2) A recursive structure may be more difficult to be implemented.

In this method, the above rule can be decomposed into the following rules. The structure of this rule set can be illustrated as in **Figure 5**.

```
1) Matching rule
   if (Source_ip == 192.168.1.*)
       Label = source_net_1;
2) Policing rules (metering rules)
   if (Label == source_net_1 &&
     Average_rate <= 1Mbps) {
     Label = source_net_1_conformant; };
   if (Label == source_net_1 &&
     Average_rate > 1Mbps) {
     Label = source_net_1_non_conformant; };
3) Marking rule
   if (Label == source_net_1_conformant) {
     DSCP = 46;          // EF
     Label = source_net_1_EF; };
4) Discarding rule
   if (Label == source_net_1_non_conformant)
     discard_all;
5) Scheduling rule
   if (Label == source_net_1_EF) {
     queue_priority = 6; };
```

In this language, types of rules are the above five. Matching, marking, and discarding rule sets are applied to a packet only once because repetitive application of these rules is unnecessary. Policing and scheduling rules are applied to a packet every time a linkage label is assigned to the packet because repetitive application of these rules is sometimes necessary. See **Figure 6**.

Flows may be aggregated by specifying the same linkage label value. For example, two flows are aggregated and marked with the same DSCP, i.e., 46, after matching in **Figure 7**.

4.2 Linkages between building blocks

A linkage label is similar to a DSCP. It is marked and tested. However, the number of linkage labels can be much larger than the number of DSCPs, i.e., 64. A linkage label is not put on a packet, i.e., it is virtual, and it is an internal state of a router.

```
      +-----------+ Label=46 +-----------+
--+-->|  Matcher  |--+------>|   Marker  |-->
  |   +-----------+  |       +-----------+
  |                  |
  |   +-----------+  |
  +-->|  Matcher  |--+
      +-----------+
```

Figure 7. Example of flow aggregation

4.3 Building block rules

The five types of building block rules are explained below.

Matching rules represent an MF or BA classifier. A matching rule tests IP packet header, i.e., one or all of DSCP, source and destination IP addresses, IP protocol, source and destination IP ports, and so on. The action of a matching rule is to assign a new linkage label to the packet.

Policing means packet discarding action that is taken when the SLA (service-level agreement) is violated. The policing function can be implemented by using a leaky bucket meter, a token bucket meter, a time window meter, or other types of meters. However, for interoperability, a metering condition should be generic and a specific type of meter should not be specified. The form of conditions is one of:

- value <= constant
- value > constant

Marking means writing a DSCP. The only condition for a marking rule is label testing. The actions are to put a DSCP and to put a new linkage label.

Discarding means packet discarding action that is taken when too many packets are queued. Discarding may occur before the queue is filled. The purpose of early discard is latency or congestion control.

Scheduling rules are used for queuing and dequeuing control. The queue priority, scheduling algorithm and its parameters can be specified in a scheduling rule. They can also be used for shaping control. The minimum and/or maximum output rate can be specified. The only condition for a scheduling rule is label testing. A hierarchical scheduling algorithm, such as class-based queueing (CBQ) can be specified by linking multiple scheduling rules.

5. A MIB/PIB for QoS Control

We designed the interface using a MIB/PIB and proposed an Internet draft to IETF. It is called the QoS PIF MIB (the Quality of Service Programming Interface Management Information Base) when it is used with SNMP, and called the QoS PIF PIB when used with COPS protocol. The definition of the MIB is given in the draft [Kan 99]. The interface language is not implemented directly

```
          source_net_1   source_net_1_conformant  source_net_1_EF
                 :                  :               :
   +-----------+ :    +-----------+ : +-----------+ : +-----------+
-->|  Matcher  |--+-->|  Policer  |-->|   Marker  |-->| Scheduler |-->
   +-----------+  |or +-----------+   +-----------+   +-----------+
                  |   +-----------+   +-----------+
                  +-->|  Policer  |-->| Discarder |
                      +-----------+ : +-----------+
                                    :
                       source_net_1_non_conformant
```

Figure 5. Connection between building block rules

```
   +-----------+                  or  +-----------+   +-----------+                  or
-->|  Matcher  |--------------------->|   Marker  |-->| Discarder |--------------------->
   +-----------+ ^                 |  +-----------+   +-----------+ ^                 |
                 |  +-----------+  |                                |  +-----------+  |
                 +--|  Policer  |--+                                +--| Scheduler |--+
                    +-----------+                                      +-----------+
```

Figure 6. Connection between building block rules

but is simulated by MIB or PIB.

5.1 Structure of the QoS PIF MIB

The QoS PIF MIB consists of four parts: the QoS-capability part, the queue-setting part, the packet-discarding-method part, and policy-rule part. The first three are defined for whole router, and the last one is defined for each logical interface. The policy-rule part simulates the rule-based language.

1) *QoS-capability part*
   This part specifies the QoS-related capability of the router. It advertises possible queuing algorithms, packet discarding algorithms, and so on to the manager. A manager, such as a policy server, may use the information from this part to determine the ability of the router. This part corresponds to a policy-rule-class (PRC) support table in the QoS PIB [Fin 99]. However, the description level of this part is lower than the PRC table and the descriptions are independent of specific policy management systems.
2) *Queue-setting part*
   This part specifies the QoS-related settings of the scheduler and the scheduling queues in the logical interfaces of the router. If an algorithm or value that is not allowed in the QoS capability part is specified, an error is returned.
3) *Packet-discarding-method part*
   This part specifies methods of discarding packets in the event of congestion. Deterministic or random early discard, or no early discard, may be specified. These methods are used in action rules that are specified in the policy-rule part. This part does not directly specify discarding actions. That packet-discarding methods are specified separately from both the rule actions and the queuing method is a distinct feature of this MIB.
4) *Policy-rule part*
   This part defines policy rules for each router interface. Classifier, meter, and action rules are separately specified. If a discarding method that is not allowed in the packet-discarding-method part is specified, an error is returned.

5.2 QoS Capability

Various routers have been developed by many vendors. The capability related to QoS control also varies. Thus, the manager of this MIB, or PIB that is converted from of this MIB, i.e., a policy server or PDP in COPS [Boy 99], must know the QoS-related capability of the router. The QoS-capability part of this MIB defines it. This part contains the set of queuing methods and packet-discarding methods that the router has. The available scheduling algorithms are explained in Section 5.3, and the available discarding methods are explained in Section 5.4.

5.3 Scheduling Methods

The scheduling-methods table consists of method definitions for scheduling packets. A scheduling method contains a scheduling algorithm, selected from the algorithms defined in the QoS-capability part of this MIB, and parameters for the algorithm. This table makes it unnecessary to specify identical scheduling methods more than once in the policy-rule part.

The available scheduling algorithms are as follows.

1) *First-in first-out (FIFO) scheduling*
   All packets are enqueued into a single queue in this algorithm. The order of dequeuing packets is exactly the same as the order of enqueuing them.
2) *Priority scheduling*
   Packets are enqueued into two or more queues according to their priority classes represented by the linkage label. Higher priority packets are always dequeued earlier.
3) *Packet-fair scheduling*
   Packets are enqueued into two or more queues according to their linkage labels. This scheduling algorithm is fair for each queue in terms of packets. Weights can be defined for each queue, if the weighted packet-fair scheduling flag is turned on in the QoS-capability part. For example, packets may be dequeued according to a (weighted) packet-by-packet round-robin algorithm when this algorithm is used.
4) *Byte-fair scheduling*
   Packets are enqueued into two or more queues according to their linkage labels. The scheduling algorithm is fair for each queue in terms of packet lengths. Weights can be defined for each queue, if the weighted byte-fair scheduling flag is turned on in the QoS capability part. For example, packet-fair scheduling is used in the fair-queuing or weighted-fair-queuing (WFQ) algorithm.
5) *Bounded byte-fair scheduling*
   Packets are enqueued into two or more queues according to their linkage labels. The scheduling algorithm is fair for each queue in terms of packet lengths. However, the bandwidth is limited for each queue. The sum of the specified bandwidths must be equal to or smaller than the bandwidth of the line. If the sum is less than the bandwidth of the line, or one or more queues do not fully use their specified bandwidth, the rest of the bandwidth may be again shared among the queues, or may be occupied by the highest-priority queue. The algorithm for using the remaining bandwidth is implementation-dependent. (The detailed algorithm may be specified in a vendor-specific MIB.) For example, bounded byte-fair scheduling will be used for implementing or simulating the class-based queuing (CBQ) [Flo 95].

If the router has two or more scheduling methods that belong to the same category above, they can be distinguished by using an additional parameter.

Queuing methods, such as WFQ or CBQ, cannot be directly specified in the scheduling-method part of the MIB, because such methods are regarded as a combination of a scheduling algorithm and higher-level control that will be implemented using other functions defined in this MIB. It is inadequate to specify a specific high-level queuing algorithm in a standardized MIB, because a very wide variety of queuing algorithms have been, and will probably continue to be, implemented in routers. The MIB should be generic.

5.4 Packet-discarding methods

The discarding-method table consists of method definitions for discarding packets. A discarding method contains a discarding algorithm, which is selected from the algorithms defined in the QoS-capability part of this MIB, and parameters for the algorithm. This table makes it unnecessary to specify the identical discarding methods more than once. Available discarding algorithms are as follows.

1) *Discarding all*
   All the packets in the virtual flow are discarded. For example, this algorithm is used when the flow is without service-level agreements (SLA) or violates an SLA. This algorithm is a special case for the deterministic early discarding (4).
2) *Tail discarding (non-early discarding)*
   Packets are discarded only when the queue is filled. This is the default algorithm and all routers must support this. Thus, no capability flag exists for this algorithm in the QoS-capability part. This algorithm is also a special case for the deterministic early discarding (4).
3) *Random early discarding (RED/WRED)*
   Packets are discarded when a specified proportion of the queue is filled. The packets to be discarded are selected at random. The proportion at which some packets start to be discarded, and the minimum proportion above which all incoming packets are discarded are specified as parameters.
4) *Deterministic early discarding (DED/WDED)*
   Packets are discarded when a specified proportion of the queue is filled. All the packets are discarded in this case, or the packets to be discarded are selected by a deterministic method. The same parameters as for RED are used in this method.

The parameters for RED and DED are defined as a value between 0 and 1000 per mil. If a parameter is not specified, its value is 1000. The discarding-method table contains a list of discarding methods available from the policy rules. Each policy rule may specify a discarding method as an action. AF PHBs can be implemented using weighted DED or RED.

5.5 Policy Rules

Policy rules here implement the building block rules defined in Section 4. Policy rules are defined using three conceptual tables: the classifier table, the metering-rule table, and the action table. The action rules defined in the action table are a combination of marking, discarding, and scheduling rules defined in Section 4.

The policy rule architecture is shown in **Figure 8**. A classifier defined in the classifier table assigns a linkage label to a packet flow. Meters defined in the metering-rule table and actions defined in the action table work only on a flow specified by the linkage label in the rules. A meter replaces the linkage label when the metering rule condition holds. If no meter is needed, the output of a classifier can be input to an action, and if more than one meters are needed, an output from a meter can be input to another meter. (The box “others” in Figure 6 actually does nothing.)

Another example, using two-stage meters is shown in **Figure 9**. The flow is classified to L1 by Classifier 1. If no metering condition is met in the first stage, L1 is not changed by “others”, and Action 1 is applied. If a metering condition is met, the flows are marked by labels L11 to L1N, and all the flows are again tested by the metering conditions. The same set of meters is applied. If no more metering conditions are met in this second stage, the linkage label is not changed by “others”, and an action from Action 11 to Action 1N is applied to each flow. If a metering condition meets, the flow is marked by one of labels L21 to L2M, and an action from Action 21 to Action 2M is applied.

A policy, i.e., a list of policy rules, is assgined to each role-combination, instead of to each logical interface, in the policy framework [Ste 99]. A role-combination can be represented by a policy table in this MIB. When a packet arrives at a logical interface, only the conditions in the classifiers linked from the rule-list table of the interface are tested.

5.5.1 Classifier table

The classifier table consists of MF and BA classifiers. If the test condition is for the protocol, the source IP address and/or port, or the destination IP address and/or port. An MF classifier can test the DS field.

MF and BA classifiers are put in the same table, unlike with the DiffServ MIB, because the order of the classifiers is sometimes significant, and because the simplest way to express the classifier order is by the order in the table. When a flow can be matched to both an MF and a BA classifiers, the flow is matched

```
   +-----------------------------------------------------------+
   |     Policy rule 1                                         |
   |    + - - - - - - - - - - - - - - - - - - - - - - - - - -+ |
   |    : +-----------+  L1  +---------+ L11 +----------+    : |
   |  +-->|Classifier1|--+-->| Meter11 |---->| Action11 |--+ : |
   |  | : +-----------+  |   +---------+     +----------+  | : |
   |  | :                |   +---------+ L12 +----------+  | : |
   |  | :                +-->| Meter12 |---->| Action12 |--+ : |
   |  | :                |   +---------+     +----------+  | : |
   |  | :                |       ...            ...        | : |
   |  | :                |   +---------+ L1  +----------+  | : |
   |  | :                +-->| others  |---->| Action1  |--+ : |
   |  | :                    +---------+     +----------+  | : |
   |  | + - - - - - - - - - - - - - - - - - - - - - - - - -|-+ |
   |  |                                                    |   |
----->|        ...               ...            ...        +------>
   |  |                                                    |   |
   |  |  Policy rule N                                     |   |
   |  | + - - - - - - - - - - - - - - - - - - - - - - - - -|-+ |
   |  | : +-----------+  LN  +---------+ LN1 +----------+  | : |
   |  +-->|ClassifierN|--+-->| MeterN1 |---->| ActionN1 |--+ : |
   |    : +-----------+  |   +---------+     +----------+  | : |
   |    :                |   +---------+ LN2 +----------+  | : |
   |    :                +-->| MeterN2 |---->| ActionN2 |--+ : |
   |    :                |   +---------+     +----------+  | : |
   |    :                |       ...            ...        | : |
   |    :                |   +---------+ LN  +----------+  | : |
   |    :                +-->| others  |---->| ActionN  |--+ : |
   |    :                    +---------+     +----------+    : |
   |    + - - - - - - - - - - - - - - - - - - - - - - - - - -+ |
   |                                                           |
   +-----------------------------------------------------------+
```

Figure 8. Policy-rule architecture

to whichever classifier is first in the table. Classifier examples are shown:

```
C1:  if (source_ip == 192.168.1.*) label = 1000;
C2:  if (DSCP == 64) label = 64;
```

A classifier assigns a linkage label to a microflow or an aggregated flow. Two or more classifiers may assign the same linkage label. Then, the conditions represented by these classifiers are “or”ed. For example, classifier C3, which cannot be expressed directly in this MIB, can be expressed by classifiers C31 and C32:

```
C3:  if (source_ip == 192.168.1.* ||
         source_ip == 192.168.3.*)
         label = 1000;    // Caonnot be expressed.

C31: if (source_ip == 192.168.1.*) label = 1000;
C32: if (source_ip == 192.168.3.*) label = 1000;
```

This notation simplifies the representation of disjunctive conditions expressed by the SMI.

There is only one classifier table in a router. The classifiers are selected and ordered for each logical interface by the policy table.

5.5.2 Metering-rule table

The metering rule-table consists of metering rules that apply to a virtual flow defined by a classifier or another metering rule, and assigns a new linkage label to a virtual flow that consists of packets matching the metering condition. The linkage label of packets that do not match the metering condition is not changed. All the entries of the metering- rule table in which the same linkage label is specified are applied sequentially (in logic) to the flow that has the linkage label.

For example, the following classifier and metering rules are given.

```
C1:  if (source_ip = 192.168.1.*) label = 1000;
M1:  if (label == 1000 && 2 Mbps < committed_rate)
         label = 1500;
M2:  if (label == 1000 && 1 Mbps < committed_rate &&
         committed_rate <= 2 Mbp)
         label = 1501;
```

The meaning of the above set of rules is equivalent to the following program.

```
if (source_ip = 192.168.1.*) {
   label = 1000;
   if (label == 1000 && bandwidth > 2 Mbps) {
      label = 1500;  … (1)
   };
   if (label == 1000 &&
      1 Mbps < committed_rate &&
      committed_rate <= 2 Mbps) {
      label = 1501; … (2)
   };
… (3)
};
```

However, the order of rules M1 and M2 is not significant, and the conditions of the meters on the same linkage label must be disjoint. The conditions of the metering rules are conjunctive. If the linkage label is not rewritten, all the metering rules for the same linkage label are tested sequentially or in parallel.

If the linkage label is updated by a metering rule, the metering rules for the updated linkage label are applied. Thus, if there are metering rules for label = 1500, they are tested at (1) in the above program.

5.5.3 Action table

The action table consists of actions that are applied to packets that have a specified linkage label. An action can be one of the following actions or a combination of them.

1) *Set a DSCP.*
   The DSCP value that is to be set to the packet is specified.

2) *Set a scheduling method and a queue number.*
   The precedence of the queue (or the queue number) in which the packet is to be put is specified.

3) *Set a discarding method.*
   The discarding method to be used for the packet is specified.

If no action is specified for a linkage label, no action is taken for a flow labeled by the linkage label.

Because the queue number and the discarding method in an action can be defined separately, different discarding methods may be specified for the same queue. If it is not possible to implement this, the SNMP agent must return an error (and may raise a trap).

Examples of actions are given.

```
A1:  if (label == 64) DSCP = 64;
A2:  if (label == 1000) {
          queuing_method = 3;
             // an index into the queuing
method table.
          queue_number = 1;
          discarding_method = 2;
             // an index into the discarding
method table.
       };
```

```
   +-----------------------------------------------------------+
   |  +-----------+                                            |
----->|Classifier1|                                            |
   |  +-----------+                                            |
   |      |L1                                                  |
   |      |                                                    |
   |      |   +--------+  L11  +--------+ L111 +---------+     |
   |      +-->|Meter11 |---+-->|Meter111|----->|Action111|--+  |
   |      |   +--------+   |   +--------+      +---------+  |  |
   |      |                |      ...             ...       |  |
   |      |                |   +--------+ L11M +---------+  |  |
   |      |                +-->|Meter11M|----->|Action11M|--+  |
   |      |                |   +--------+      +---------+  |  |
   |      |                |   +--------+  L11 +---------+  |  |
   |      |                +-->| Others |----->|Action11 |--+  |
   |      |                    +--------+      +---------+  |  |
   |      |      ...                                        |  |
   |      |                                                 |  |
   |      |   +--------+  L1N  +--------+ L1N1 +---------+  |  |
   |      +-->|Meter1N |---+-->|Meter1N1|----->|Action1N1|--+  |
   |      |   +--------+   |   +--------+      +---------+  |  |
   |      |                |      ...             ...       |  |
   |      |                |   +--------+ L1NL +---------+  |  |
   |      |                +-->|Meter1NL|----->|Action1NL|--+  |
   |      |                |   +--------+      +---------+  |  |
   |      |                |   +--------+ L1N  +---------+  |  |
   |      |                +-->| Others |----->|Action1N |--+  |
   |      |                    +--------+      +---------+  |  |
   |      |                                                 |  |
   |      |   +--------+ L1             L1     +---------+  |  |
   |      +-->| Others |---------------------->|Action1  |--+----->
   |          +--------+                       +---------+     |
   |                                                           |
   |    ...                                                    |
   |                                                           |
   +-----------------------------------------------------------+
```

Figure 9. Example of an architecture with two-stage meters

6. Concluding Remarks

A method of expressing a policy rule by a rule-based language using building block rules and linkage labels was proposed in this paper. The outline of the language was described, and a MIB/PIB that simulates the language was described too. The two design goals of the language were interoperability and syntagmatic modularity. The latter has been mostly achieved by free combination of building block rules. However, the former depends on the detail of the language design, and both the language nor the QoS PIF MIB/PIB have not yet been implemented. Only the preliminary version of QoS PIF PIB was implemented.

Future work include detailed specification of the rule-based language and implementation of the language on routers of several vendors, including Hitachi's giga-bit router GR2000.

Acknowledgment

We thank Takeshi Aimoto and Takeki Yazaki from Hitachi Japan for discussion on low-level router modeling, and thank Spyros Denazis from Hitachi Europe and Satoshi Yoshizawa form Hitachi America for intensive discussion on high-level router modeling.

References

[Bak 99] F. Baker, "Management Information Base for the Differentiated Services Architecture", draft-ietf-diffserv-mib-00.txt, *Internet Draft*, July 1999.

[Ber 99a] Y. Bernet, et al, "A Framework for Differentiated Services", draft-ietf-diffserv-framework-02.txt, *Internet Draft*, February 1999.

[Ber 99b] Y. Bernet, A. Smith, and S. Blake, "A Conceptual Model for Diffserv Routers", draft-ietf-diffserv-model-01.txt, *Internet Draft*, October 1999.

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

[Boy 99] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry, "The COPS (Common Open Policy Service) Protocol", draft-ietf-rap-cops-08.txt, *Internet Draft*, November 1999.

[Bra 99] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

[Cas 99] J. Case, R. Mundy, D. Partain, and B. Stewart, "Introduction to Version 3 of the Internet-standard Network Management Framework", RFC 2570, April 1999.

[Cov 99] S. Covaci, ed., "Active Networks", *Lecture Notes in Computer Science*, Springer-Verlag, June 1999.

[Fin 99] M. Fine, K. McCloghrie, S. Hahn, K. Chan, and A. Smith, "An Initial Quality of Service Policy Information Base", draft-mfine-cops-pib-02.txt, *Internet Draft*, October 1999.

[Flo 95] S. Floyd, and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks", *IEEE/ACM Transactions on Networking*, Vol. 3 No. 4, pp. 365-386, August 1995.

[Har 99] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", RFC 2571, April 1999.

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

[ISO 87] "Information processing systems — Open Systems, Interconnection — Specification of Abstract Syntax Notation One (ASN.1)", International Standard 8824, *International Organization for Standardization*, December 1987.

[Kan 99] Y. Kanada, M. Ikezawa, S. Miyake, and Y. Atarashi, "SNMP-based QoS Programming Interface MIB for Routers", draft-kanada-diffserv-qospifmib-00.txt, *Internet Draft*, October 1999.

[McC 99a] K. McCloghrie, D. Perkins, and J. Schoenwaelder, ed., "Structure of Management Information Version 2 (SMIv2)", RFC 2578, April 1999.

[McC 99b] K. McCloghrie, D. Perkins, J. Schoenwaelder, J. Case, M. Rose, and S. Waldbusser, "Textual Conventions for SMIv2", RFC 2579, STD 58, April 1999.

[McC 99c] K. McCloghrie, D. Perkins, J. Schoenwaelder, J. Case, M. Rose, and S. Waldbusser, "Conformance Statements for SMIv2", RFC 2580, STD 58, April 1999.

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

[OMG 98] "The Common Object Request Broker: Architecture and Specification", Revision 2.2, *Object Management Group, Inc.*, February 1998.

[She 97] S. Shenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, September 1997.

[Ste 99] M. Stevens, W. Weiss, H. Mahon, B. Moore, J. Strassner, G. Waters, A. Westerinen, and J. Wheeler, "Policy Framework", draft-ietf-policy-framework-00.txt, *Internet Draft*, September 1999.

[Wro 97] J. Wroclawski, "Specification of the Controlled-Load Network Element Service", RFC 2211, September 1997.