QoS Policy Modeling for Conflict Detection

Taghrid Samak
School of Computer Science, Telecommunication,
and Information Systems
DePaul University
Chicago, Illinois 60604
taghrid@cs.depaul.edu

April 16, 2007

Abstract

Policy-based network management is a necessity in large scale management environment. It provides means for separating high level system requirements from actual implementations. As the network size increases, the need for automatic tools to perform management increases rapidly. In Differentiated Services (DiffServ), policies can be used to dynamically reconfigure routers such that the desired Quality of Service (QoS) goals are achieved as well as to perform admission control. Despite its potential benefits, policy-based management is far from being widely adopted. One of the reasons behind this is that it is difficult to analyze policies in order to guarantee configuration stability. Policies configured on large domains may have conflicts leading to unpredictable effects. We propose developing a formalization to put these policies in canonical form that can then aid in detecting and solving conflicts in these policies configurations. We also present a classification of these conflicts with respect to their scope and data types involved. Moreover, we describe a simple method of analyzing the severity of the conflict based on tunable user sensitivity.

1 Introduction

Network management is almost always specified in user given policies that are used to set everything in the network behavior; routing maps, allowable traffic into and out of the network, and traffic/flow quality specifications. Therefore, policy-based network management is a necessity in large scale management environment. It provides means for separating high level system requirements from actual implementations. As the network size increases, the need for automatic tools to perform management increases rapidly. The main focus of this work is Quality of Service (QoS) policies. In Differentiated Services (DiffServ), policies can be used to dynamically reconfigure routers such that the desired QoS goals are achieved as well as to perform admission control. Developing and deploying a complete policy-based management system for QoS is still an ongoing problem. Guaranteeing configuration stability and correctness is a major issue. Configuring policies on large domains might cause conflict between policy parameters within different devices in the domain. Those conflict can lead to performance instability, unpredictability and degradation.

We propose developing a formalization to put these policies in canonical form that can then aid in detecting and solving conflicts that might occur in these policies. In order to successfully discover these conflicts, we have to find a reasonable classification with respect to their scope (*i.e.*, flow, device, or network wide) and data types involved (*i.e.*, Boolean, ordinal, etc). In addition, we describe a way to model the policy parameters into a unified representation to facilitate further analysis and processing. Moreover, we describe a simple method of analyzing the severity of the conflict based on tunable user sensitivity.

We will start by classifying and modeling the parameters affecting each per-hop-behavior (PHB). This classification will help us understand parame-

ters interactions and how this will affect actions involving those parameters. Depending on this classification, the conflicts between actions can be analyzed. Then, each of the parameters will be encoded into the unified representation that will use Binary Decision Diagrams (BDDs). The analysis will first be discussed on a per-flow basis. Afterwards, we will present how all flow-class requirements mentioned at all devices in the domain can be incorporated in a single model to avoid the inevitable processing explosion that will take place if we implemented the system to analyze one flow at a time.

In summary, we propose a bottom-up approach to policy representation and conflict detection and analysis. We first start by defining policy parameters that govern action performed on each flow. The parameters aggregated together form a single per-hop behavior (PHB) associated to that flow. Aggregation of consecutive PHBs for that flow forms the overall per-domain behavior (PDB) associated to this specific flow.

The main advantage of this work over other QoS conflicts detection techniques is the unified canonical form for representing policies. Also, per-domain behavior analysis has not been addressed in previous work. Only policy conflicts within a single Diffserv device (single PHB).

The paper is organized as follows. First we survey techniques used for QoS DiffServ policy conflicts analysis. We compare and contrast our approach with those previous techniques. In section 3, we present the proposed policy modeling procedure. Bottom-up representations are described from singe parameter flow level to domain level. In sections 4 and 5, conflicts classification is proposed. The detection and analysis are introduced in section 6. Section 7 shows the application of our classifications to standard QoS policies. We conclude the discussion in section 8

2 Related Work

Most of the work done in the area considered policy-based management in specific domain, and restricted to a certain framework. One of the most popular frameworks in Europe is the TEQUILA framework, introduced in [3]. Static conflicts are introduced in [2]. A detection mechanism is applied to the network dimensioning part of the TEQUILA framework. Since static conflicts are analyzed before applying the actual policy, the administrator is required to perform changes to resolve conflicts. In [1], analysis of dynamic conflicts was proposed. This work also introduces domain specific dynamic conflicts detection and resolution in the context of the TEQUILA framework.

A general classification for conflicts in policy-based distributed system management was introduced in [5]. This general taxonomy has been used in different domain to help classify conflicts depending on the functionality and parameters. QoS policy conflicts classification was introduced in [2] and [1] considering TEQUILA framework. Those policies focused on per-device conflicts, conflict residing on a single machine. The classification also was limited to a set of predefined PHBs. In this work we propose a domain-wide conflict analysis independent from specific behaviors. This canonical representation can be extended to other applications.

The representation proposed here is based on the one used in [4] for security policy modeling. QoS policies are more general than security policies in terms of the number of actions allowed and the parameters affecting actions.

3 Policy Modeling and Encoding

We have several levels of abstractions in this model. First, there is the per-flow PHB. This describes the properties and parameters enforced on a

specified flow at a single network node. Second, there is the overall PHB affecting a specified flow when considered over all nodes of the domain. Finally, we consider combining all the flows specified in policies at different nodes, to have a collective view of the domain policy.

The encoding of this information will take place in a Boolean expression format that will encode all the criteria and policy values in the form of the expression. This will be implemented in turn in BDDs to facilitate operations and counter example searching.

3.1 Per-flow PHB Modeling

To be able to analyze policies using this model, each PHB will be represented as a Boolean function. A PHB consists of many parameters. We will consider the PHB specification from QoS policy Information Model [6]. Each PHB has two main categories of actions; actions controlling bandwidth and delay, and congestion control actions. Bandwidth and delay actions are responsible for enforcing fairness between different classes. A complete mapping of all the parameters is discussed in section 7.

The first step towards conflicts analysis in DiffServ environment is to model each PHB in a way that will simplify and facilitate action representation and modeling. The following types of parameters inclusively characterize each PHB:

- *Boolean:* A boolean parameter is a bit value that will be either $TRUE$ or $FALSE$. Conflicts for those parameters can be identified by a simple equality operator. An example of this type is the fairness field defined to enforce fairness in the bandwidth or delay assignment.

- *Quantitative:* A quantitative parameter is a numerical value that describes a certain value assigned to control the forwarding behavior. A

quantitative parameter could be maxDelay, maxJitter, queue size or maxPacketSize. This value will be encoded using a number of bits depending on its range of values. In other words the number of bits $B = \log(\lceil N \rceil)$, where N is the maximum value this parameter can have. The Boolean variables assigned to such a parameter will be included in the PHB expression in its positive or negative form depending on the binary encoding of the parameter.

- *Range:* This type of parameters describes ranges of values allowed for a certain property of a PHB. A range parameter is defined as minValue and maxValue. Bandwidth can be classified as a range parameter. However, storing the exact min and max values is not needed. The overall range can be encoded by the disjunction of all the values in the range where each is presented as a single quantitative parameter as mentioned above. Thus, a single expression over $\lceil \log(N) \rceil$ variables will hold the minimum and maximum criteria of the parameter.

A PHB action is a combination of all those parameters values.

A Boolean expression representing a single PHB will have the following number of bits:

$$L = \mid V_B \mid + \sum_{v_i \in V_Q} \lceil log_2(max(v_i)) \rceil + \sum_{v_i \in V_R} \lceil log_2(max(v_i)) \rceil \qquad (1)$$

where V_B is the set of Boolean parameters, and V_Q is the set of quantitative parameters, V_R is the set of range parameters, and $max(v_i)$ is the maximum possible value of parameter v_i.

A single PHB response to a certain flow is the combination of all parameters involved in the policy definition with respect to that flow. For a node

i, the PHB corresponding to traffic flow j can be evaluated as:

$$\begin{aligned} PHB_i^j \equiv PHB_{b_1}^j | PHB_{b_2}^j | \ldots | PHB_{b_n}^j | \\ PHB_{q_1}^j | PHB_{q_2}^j | \ldots | PHB_{q_m}^j | \\ PHB_{r_1}^j | PHB_{r_2}^j | \ldots | PHB_{r_k}^j \end{aligned} \tag{2}$$

where $b_i \in V_B$, $q_i \in V_Q$ and $r_i \in V_R$ correspond to all boolean, quantitative and range parameters respectively. The | operator is the bit concatenation between all binary representations of PHB parameters.

3.2 Per-flow Overall Modeling

For each parameter type, T, we have an aggregation operator, $\bowtie_T$, that controls the final action (QoS guarantees) for this parameter type, where $T \in \{B, Q, R\}$ corresponding to boolean, quantitative and range respectively.

Now, for each flow, j, we calculate the per-domain behavior, PDB_T^j, for parameter, T.

$$PDB_T^j \equiv PHB_1^j \bowtie_T PHB_2^j \bowtie_T \ldots \bowtie_T PHB_n^j \tag{3}$$

where $1, 2, \ldots, n$ are the nodes on the path of flow j. The operators for each parameter will be discussed in more detail with conflicts in section 6.

The overall relation for all parameters affecting PDB for a flow, j can be formulated as:

$$\begin{aligned} PDB^j \equiv PDB_{b_1}^j | PDB_{b_2}^j | \ldots | PDB_{b_n}^j | \\ PDB_{q_1}^j | PDB_{q_2}^j | \ldots | PDB_{q_m}^j | \\ PDB_{r_1}^j | PDB_{r_2}^j | \ldots | PDB_{r_k}^j \end{aligned} \tag{4}$$

where $b_i \in V_B$, $q_i \in V_Q$ and $r_i \in V_R$ correspond to all boolean, quantitative and range parameters respectively. The | operator is the bit concatenation between all parameters.

To summarize, each flow treatment will be calculated based on the defined policy at each node with respect to this flow. For each parameter the aggregation is evaluated using the parameter operator. PDBs for each parameter depends on the individual PHBs for that parameter. The overall PDB for the flow is formed by combining all parameters behaviors.

3.3 Overall Network Modeling

In this step, the filtering condition applied to each flow is formalized. We can view this part as the overall policy applied to the domain. The policy consists of a set of rules. Each rule has a condition and an action. The condition filters incoming traffic, and triggers the action for matched traffic. The action over all domain routers corresponds to the per-domain behavior for the traffic class matching the condition. Each rule is defined as:

$$R_i := C_i \Rightarrow PDB_i \tag{5}$$

Each rule, R_i, maps traffic matching condition, C_i, to a specific PDB_i. PDBs can be found from equation 4. Packets arriving are matched in-order against each rule. The first rule that the packet satisfies its condition is triggered. The behavior associated to the triggered rule is applied to the packet.

Assume that the rule triggered when a packet from traffic class j arrives is R_i. The overall satisfied condition resulting from this matching will be:

$$C^j = \neg C_1 \wedge \neg C_2 \ldots \wedge \neg C_{j-1} \wedge C_j \tag{6}$$

The overall filtering condition that will be applied on all flows can be written as:

$$C = C^1 \vee C^2 \ldots \vee C^n \tag{7}$$

where n is the total number of traffic classes.

The overall filtering policy that will be applied to a packet p, can be formalized as:

$$Q_{PDB_i} = (p \wedge C) \Rightarrow R_i \tag{8}$$

where p is a the binary representation of packet header fields. The conditions are identified by defining values for all or some header fields.

4 Conflict Scopes

Conflicts in QoS policies can appear at several scopes. Following are the different scopes where such conflicts can occur.

- *Intra-PHB Conflicts*
  These include conflicts that occur within the flow properties at a specific node.
  - *Single Parameter Conflict*
    These are the conflicts that occur due to malformed parameter conditions. For example, by specifying negative Queue length, or the minimum of a variable is greater than its maximum range, or having a percentile parameter greater than a hundred.
  - *Multi-Parameter Conflict*
    These conflicts take place between different parameters. They are more complex to identify, and resolve. For example, if a priority level is specified by a flow, then the maximum bandwidth should

be specified, otherwise starvation for other flows will take place.

- *Inter-PHB Conflicts*
  The hardest conflicts to detect and resolve are those that take place due to policy definitions across different nodes (even across domains). Inter-PHB conflicts happen when the same flow meets different behavior from more than one node on its way from source to destination. PHB policy are set such that a flow meets some quality requirements as needed by the end user, and this needs that the flow meets equivalent treatment at all hop from end to end along its path. If this homogeneity is not available, the flow will have its quality reduced as the worst PHB settings it meets. Checking the conflicts for a certain flow, requires extracting its PHB along the path, and comparing them together for conformity. An example of these conflicts can be different bandwidth allocations at different nodes, or type of forwarding priority at successive nodes.

- *Cross-flow Intra-Node Conflicts*
  Conflicts between different flows in the same node are orthogonal to conflicts along the path of the flow. Changes to correct one conflict can resonate in the other class of conflicts, thus several iterations might be needed. These conflicts take place when the settings that are assigned to each flow at a specific node are not feasible to satisfy simultaneously. For example, assigning a minimum bandwidth for each flow, such that the sum is more than the link capacity. Such settings are valid if observed on the flow level, but problems arises when we consider the inter-flow effect.

- *Cross-flow Inter-Node Conflicts*
  More complex problems arises when the settings of flows can affect each

other in more than one node. In most cases, they are incorporated with routing settings, and device states. These conflicts need further investigation to classify and analyze. They are out of the scope of this work, to be studied in future research.

5 Conflict Strength

Conflicts classification according to scopes is one possible way for the classification. Another orthogonal classification on conflicts is according to strength. We can view this as a measure of how a conflict affects network conditions. If the conflict will prevent normal operation of the network, it is an intolerable conflict. Here, we introduce the concept of conflict strength or, degree. Since most of QoS parameters and properties have fuzzy nature, the network condition may vary depending on the type and degree of the conflict, or conflicting parameters.

We distinguish here between two types of conflicts; hard conflicts, and fuzzy conflicts.

5.1 Hard Conflict

This type of conflict corresponds to the assignment of different values to a single parameter. If the behavior associated with a certain traffic is different for multiple nodes within the domain then, the policy is in conflict. No parameter or behavior is allowed to have different values for a certain traffic class. The policy needs to be reconfigured to resolve this conflict. We call is *Hard* since any slight variation in any of policy parameters will trigger a conflict.

5.2 Fuzzy Conflict

Fuzziness in conflicts comes from behaviors that differ between nodes on the network with respect to the same flow. The values for a specific parameter might differ, but the service can still be achieved. It will be unacceptable to reject flows just because the bandwidth limitations on different routers are not the same. A compensation can be allowed to grant some service given different nodes' configurations.

Here we introduce a conflict measure that will be used for all detection steps in the coming sections. Figure 1 shows a proposed conflict level estimation scheme, $CL - chart$. For each parameter, or conflicting configuration, the conflict level can be estimated from different policies. On the $x - axis$, a measure on the parameter will be calculated (the deviation of the values between different nodes). For this value, the corresponding conflict level, $y - axis$, represents the strength of this conflict. The conflict level, CL, is a value in the interval $[0, 1]$. If $CL = 0$, then there is no conflict detected. The conflict level will increase until it reaches the maximum value $CL = 1$, which the network cannot tolerate. In between, the slope of the line will depend on the specific conflict description and the parameter involved.

This graph will represent the relation between conflict and policy configuration parameters in the case of quantitative and range parameters. An example for this is the queue length. If the queue length assigned to a specific flow differs over the path the flow will traverse, the overall performance of this flow will be limited to the minimum length over the path. In this case, the variation of the lengths will affect the conflict level.

The curve characteristics can be modified according to the parameter. It can be unified over the same class of parameters or may be determined for each PHB property. In section 6, we propose measures ($x - axis$) for quantitative and range parameters. The CL value can be determined depending

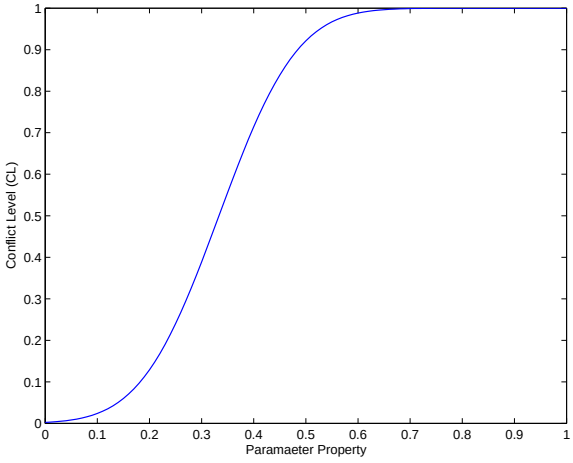

Figure 1: Conflict Level Chart

on the system.

6 Conflict Analysis and Detection

This section will deal with techniques to discover different kinds/scopes of conflicts. As discussed in previous sections, we have scope conflicts and strength conflicts. Each conflict discussed here will be mapped to one of the previously mentioned types.

6.1 Intra-PHB Conflict

The simplest form of conflicts are those that concern a single flow and a single PHB policy. As mentioned earlier, they are further classified to single parameter and cross-parameter conflicts. The first kind is detected at the parsing stage, when a user-provided policy fails to pass the first checking phase that leads to encoding. Using the same example mentioned earlier, a

negative value for the queue length will raise a flag that it is not possible to be encoded. Therefore, we claim that all such conflicts will be discovered in this early phase.

Once parameters are encoded individually, incorporating them into a single expression (still for a specific flow at a certain node) will show if any problems between parameters will take place. Using the power of our representation as Boolean expression we can add all kinds of constraints into our expression. For example, if a condition specifies that parameter p_1 cannot have a $false$ value, when another parameter p_2 is greater than or equal 4. This can be checked as follows: Assume the parameters are assigned variables x_1, and $x_2x_3x_4$, then by multiplying the resulting expression from the single parameter encoding phase by $\neg x_1x_4$ will result in an overall value of $false$ if the conflict exists.

6.2 Inter-PHB Conflict

To determine the final treatment of a certain flow within a domain, per-domain behavior PDB, we need to examine all PHBs along the path. For each type of parameter, a different operator will control the aggregation of the PHBs. We assume here that all parameters are mentioned in the policy. If a high level representation does not provide a parameter value, the default on the machine will be used.

An inter-PHB conflict occurs when the final aggregation of all PHBs for each traffic class is not consistent with the required service.

Now, we propose possible operators that can be used to evaluate the aggregate PHB. A conflict rises when the resulting behavior is different from the expected one, or the aggregation resulted in an empty behavior.

6.2.1 Boolean Parameters Aggregation

For a boolean variable, a simple equivalence of the values for the parameter at each node will be the final behavior. Since boolean parameters take only two values $TRUE$ or $FALSE$, the equivalence will be $TRUE$ if all PHBs have the same value for the parameter.

For a boolean parameter, the operation $\bowtie_B$ will be the logical equivalence of the values. For this conflict to happen at parameter b, for the traffic class j, the following condition muse be $TRUE$:

$$PHB^j_{b_l} \equiv PHB^j_{b_k} \ \forall\, l, k \tag{9}$$

where l, k are routers within the domain.
This conflict is a hard conflict, it either exists or does not.

6.2.2 Quantitative Parameters Aggregation

Most of the quantitative parameters in a single PHB correspond to resource allocation for a certain flow. Examples of those parameters are queue length, priority, maxDelay and maxJitter. Since those parameters depend on the physical capabilities of routers on the domain, the most suitable way to aggregate them is to take the minimum. In this case, we limit the resources used to serve this traffic class to the minimum allowed by the domain. BDDs allow the evaluation of the minimum and maximum using boolean operations within a fixed time.

Conflicts for quantitative parameters are fuzzy. We can guarantee limited resources according to network and device conditions. Here, we need to define the characteristics of the Conflict Level chart, $CL - chart$, described in section 5. Depending on all PHBs values for this flow, a measure of deviation between them can be calculated. This measure is used to locate

the point on the $CL - chart$ that characterize the conflict. Queue size for example might have a value of 5 on one PHB, and 50 on another PHB for the same traffic. The value returned by the aggregation along the path will be 5, which may not acceptable for some PHBs. This should raise a conflict in the policy.

To be more specific, we need to evaluate how far the resulting aggregate PHB for this parameter is from the majority of the requests. For the queue length example, if the device majority has queue length close to the minimum value, 5, then this should not be a conflict. On the other hand, the majority might have a big value gap from the minimum, resulting in a conflict.

So, for quantitative parameters, the $x - axis$ of the $CL - chart$ should correspond to the deviation of the parameter values. Assume we have n nodes, and the value of the parameter is x_i at each node. First, we need to calculate the statistical *mode* of the values (the value that has the maximum probability). In our case, it will be the resource allocation value that is present at the majority of the nodes. The deviation can then be calculated as:

$$dev(x) = \begin{cases} \frac{mode(x) - min(x)}{mode(x)} & p(x = mode(x)) > 0.5 \\ \frac{max(x) - min(x)}{max(x)} & o.w. \end{cases} \quad (10)$$

The first part is when there is a dominating common value of x. In this case, the deviation should be relative to this common value, since most of the devices can afford it. The latter case is when there is no majority value. In this case, the deviation between the maximum and minimum values is the one to consider. As the value of $dev(x)$ increases, the conflict level CL increases. It reaches the maximum 1 when the minimum is *zero*. The minimum value of conflict, $CL = 0$, is achieved when $min(x) = max(x)$. This corresponds to the fact that all devices have the same parameter value, hence no conflicts.

6.2.3 Range Parameters Aggregation

Representing ranges as binary numbers simplifies the aggregation process of such parameters. Aggregation across different PHBs can be performed by taking the intersections of the available ranges. A logical AND operator will be able to capture this intersection interval. If the conjunction results in $FALSE$, then there is no possible sub-range to be assigned to this parameter. This will constitute the definite conflict, $CL = 1$.

This type of conflict belongs also to the fuzzy classification. We need to identify the $x - axis$ values on the $CL - chart$, to be able to asses the degree of the conflict. The following expression calculates the deviation of the values using logical conjunction and disjunction operations.

$$dev(x) = 1 - \frac{|\bigwedge_{i=1..n} x_i|}{|\bigvee_{i=1..n} x_i|} \tag{11}$$

where x_i refers to the boolean expression resulting after mapping each parameter value. The numerator is the number of satisfying assignments representing the common range. The denominator corresponds to the total number of satisfying assignment for the union of all values. In other words, we want the conflict level to reflect the relative area of the intersection (common sub-range) over the area of the union (the widest range). Using BDD encoding of variables provide the capability of counting satisfying assignments efficiently.

7 Representation Completeness

In this section, we show how the proposed classification maps all PHB parameters defined in QoS policies. We also show that any new actions or conditions can be easily represented using the proposed approach.

Class	Property	Parameter Type	Representation
PHB	Max Packet size	Quantitative	Value
Bandwidth	Forwarding priority	Quantitative	Value
	Bandwidth units	Boolean	Multi-bits
	Min Bandwidth	Quantitative	Range
	Max Bandwidth	Quantitative	Range
	Max Delay	Quantitative	Value
	Max Jitter	Quantitative	Value
	Fairness	Boolean	Single bit
Congestion Control	Queue size units	Boolean	Multi-bits
	Queue size	Quantitative	Value
	Drop method	Boolean	Multi-bits
	Drop Threshold units	Boolean	Multi-bits
	Drop Threshold method	Boolean	Multi-bits
	Min Threshold value	Quantitative	Range
	Max Threshold value	Quantitative	Range

Table 1: QPIM PHB properties mapping

We present action parameters as specified in the Policy QoS Information Model, [6]. PHB properties are divided into three classes in this model; QoSPolicyPHBAction, QoSPolicyBandwidthAction and QoSPolicyCongestionControlAction. Each class has a set of properties. We aim here to map all those properties to the specific parameter type that can accommodate the property. We show here that all class properties can be handled by our representation. In table 1 each class along with its properties are mapped to parameter type (boolean, quantitative or range). Parameter type entry corresponds to the initial classification of the parameter. The last column shows the actual representation that will be stored in the BDD. Some boolean types have multiple bits corresponding to different options. For example, bandwidth unit could be a percentage, or an absolute value. The property is mapped to two bits in the final BDD.

8 Conclusion and Future Directions

In this work, we proposed a novel representation of QoS policy parameters for DiffServ networks. The representation is based on Binary Decision Diagrams. This canonical form simplifies conflict detection and analysis by performing only binary operations on the specified policies. We followed a bottom-up approach, considering traffic classes/flows. First, possible parameters controlling a policy PHB are classified and clustered into three main types (Boolean, Quantitative and Ranges). A flow-specific PHB is then formed using policy parameters. The aggregation of PHBs affecting a specific flow through the domain is then calculated. At each step/scope, conflicts can be analyzed. We also introduced the notion of conflict level. Depending on different parameter values on different PHBs affecting a certain flow, a conflict level CL can be used to asses the severity of the mis-configuration. The detection of per-domain behavior conflicts is proposed based on the CL values. We then showed that the parameters classification proposed here can accommodate all possible PHB properties.

As future directions, we will explore the applicability of our approach to other QoS policy parameters, not only PHBs. Marking, shaping and policing properties can also be encoded in the same manner.

The integration with a high level policy representation will be investigated to facilitate the testing procedure.

Some conflicts cannot be addressed unless explicitly states. Such conflicts include the dependencies between parameters (Intra-PHB, Multi-parameter conflict). A generalization for these dependencies need to be studied.

Cross-flow conflicts need to be investigated as well, since it lies within the PHB of the node. We considered here only policy applied to a single flow, over the domain.

References

[1] Marinos Charalambides, Paris Flegkas, George Pavlou, Arosha Bandara, Naranker Dulay, Emil Lupu, Javier Rubio-Loyola, Alessandra Russo, and Morris Sloman. Dynamic Policy Analysis and Conflict Resolution for DiffServ Quality of Service Management. In *IFIP/IEEE Network Operations and Management Symposium (NOMS 2006)*, April 2006.

[2] Marinos Charalambides, Paris Flegkas, George Pavlou, Arosha Bandara, Emil Lupu, Alessandra Russo, Naranker Dulay, Morris Sloman, and Javier Rubio-Loyola. Policy Conflict Analysis for Quality of Service Management (2005). In *6th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2005)*, June 2005.

[3] P. Flegkas, P. Trimintzios, and G. Pavlou. *IEEE Netwrok*.

[4] Hazem Hamed, Ehab Al-Shaer, and Will Marrero. Modeling and verification of ipsec and vpn security policies. In *ICNP '05: Proceedings of the 13TH IEEE International Conference on Network Protocols (ICNP'05)*, pages 259–278, Washington, DC, USA, 2005. IEEE Computer Society.

[5] Emil C. Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering*, 25(6):852–869, November/December 1999.

[6] Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, and B. Moore. Policy Quality of Service (QoS) Information Model. RFC 3644 (Proposed Standard), November 2003.