

QoS Policy Modeling for Conflict Analysis

Taghrid Samak, Ehab Al-Shaer, Hong Li
School of Computer Science Intel IT
DePaul University

Email: {taghrid,ehab}@cs.depaul.edu hong.c.li@intel.com

Abstract—Policy-based network management is a necessity in managing large scale environments. It provides means for separating high level system requirements from actual implementation. As the network size increases the need for automated tools to perform management increases rapidly. Configuring routers and network devices to achieve Quality of Service goals is a challenging task. Using Differentiated Services to dynamically perform this configuration involves defining policies on different network nodes along multiple domains. Policy aggregation across domains requires a unified policy model that can overcome the challenge of conflict detection and resolution. In this work, we propose a unified model to represent and encode QoS policies that will result in efficient and flexible conflict analysis ability. The representation utilizes a bottom-up approach, from the base policy parameters to the aggregation of overall policy across domains with respect to traffic classes. We also present a classification of these conflicts and a simple analysis method for the severity of the conflict.

I. 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 re-configure 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 conflicts between policy parameters within different devices in the domain. Those conflicts can lead to performance instability, unpredictability and degradation. Inconsistent configurations for different policies that handle the same traffic class can lead to unsatisfied overall performance and violation to service level agreements. On the other hand, those conflicts not only affect the quality of service, they may also have security impacts. Compromised nodes will have misconfigurations that can affect the overall performance of the network.

In this work we try to address conflicts from a new point of view. A certain flow might have different per-hop behaviors (PHBs) on its path from source to destination. Conventional

analysis techniques announce those differences as conflicts and halts network operations until manual resolution is applied. In this work, we try to find a way to bypass those differences without interrupting the system. A flow suffering from different PHBs can still pass through the domain. The concept of a fuzzy conflict is introduced to overcome previous hard analysis.

We will start by classifying and modeling the parameters affecting each per-hop-behavior (PHB). This classification will help us understand parameters 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 per-domain behavior (PDB) to avoid the inevitable processing explosion that will take place if we implemented the system to analyze one flow at a time.

We first propose a new representation of QoS policies based on per-flow analysis, starting from flow passing a single device to the overall domain behavior. The overall process is depicted in figure 1. Each policy is defined as actions corresponding to each traffic class/flow (PHB for this class). The PHB encoders produces the new representation of the policies, and perform conflict analysis for each PHB. The resulting conflict-free PHB policies are then fed to the domain analyzer, where the analysis is performed on a parameter basis. The overall result of this process is an aggregated "conflict-free" policy for the domain.

In this approach conflict analysis is performed in two layers: PHB-analysis which is then aggregated to form the PD policy. The first layer is required to produce an absolute conflict-free policy, in order to be able to use the results through the second layer. In the overall domain analysis, we introduce the concept of conflict strength. This will be the means of bypassing inconsistency in the configuration according to the severity of the detected conflicts.

In summary, we propose a bottom-up approach to QoS policy representation and conflict detection and analysis. We first start by defining policy parameters that govern actions 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. In this way, identifying policy rules that cause conflicts

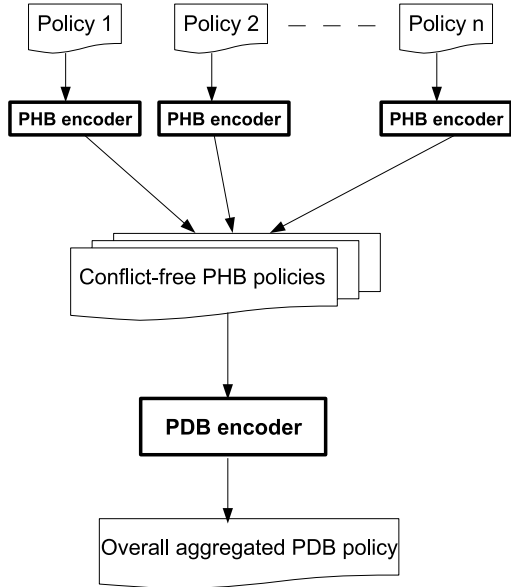


Fig. 1. PDB Analysis

for specific flows will be much easier. High level analysis can be incorporated to help resolve conflicts.

The main contributions of this work are:

- Using policy actions' parameters in a unified model for representing policies (can be used with any existing high level provisioning).
- Policy conflicts analysis for single PHB.
- Policy aggregation and conflict analysis for multiple PHB (PDB).
- Introducing a generic measure for conflicts (conflict level) that will help assess the inconsistency in policy configuration.

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 III, we present the proposed policy modeling and action parameters model. Two classifications for policy conflicts are also discussed. In the following two sections the bottom-up representations are described from single parameter flow level to domain level. In section IV, PHB modeling is presented along with conflict analysis for that level. Section V presents the model for PDB analysis, and its conflicts. Through out the discussion examples are provided for policy conflicts and their resolution strategy. A simple case study for the approach is presented in section VI. We conclude the discussion in section VII

II. 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 [7]. Static conflicts

are introduced in [4]. 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 [3], 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 [9]. This general taxonomy has been used in different domains to help classify conflicts depending on the functionality and parameters. QoS policy conflicts classification was introduced in [4] and [3] 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.

The same classification of conflicts is used in both papers. The first distinction is between domain independent conflicts and conflicts for specific applications. The first category contains a straight forward analysis of policy conditions. The conflicts include rule redundancy, and action mutual exclusion. Both can be detected by a simple conjunction of the involved policy fields. For application-specific conflicts, the work is approached depending on the module responsible for the policy definition. For the TEQUILA framework, network dimensioning handles static policies, while the resource management module is responsible for system dynamics and runtime analysis.

In [4] static conflicts were considered, where no resolution is performed by the system. The administrator has to modify the policy to resolve those conflicts. Conflicts of this type include bandwidth assignment conflicts and routing conflicts. This work corresponds to our analysis of hard conflicts.

Dynamic conflicts analysis, addressed in [3], deals with conflicts arising from current state of the system. Those conflicts occur due to either inconsistencies between statically defined policies and system state, or interactions between different modules in the system. A dynamic resolution of the conflicts was proposed. This part corresponds to our fuzzy analysis of conflicts.

Our approach is different from the previous analysis in the policy representation part. We propose a canonical representation for policies, that can be deduced from any definition. We also propose a low level representation of conflicts that generalizes to any policy parameters. Our new representation enables dealing with traffic flows as independent entities. This will help identify suffering classes, and hence updating the policy will be much easier.

The representation proposed here is inspired from the one used in [8] and [1] for security policy modeling. In this work, Binary Decision Diagrams (BDDs), [2], were used to model security policies. The domain of QoS policies is more general than security policies in terms of the number of actions allowed and the parameters affecting actions. In security policies, simple logical operations were sufficient to

analyze conflicts. In our case we use more complex operations. BDDs enable the evaluation of this complicated analysis in an efficient way.

Fuzzy logic has been used for resource provisioning in QoS, [6] and [5]. Those approaches were applied on policies assuming no conflicts, and the main goal is to interact with the current state of the network. In our approach we use a fuzzy measure to assess policy conflicts between devices.

III. GENERAL POLICY MODEL AND CONFLICT CLASSIFICATION

QoS policy defined over a set of flow classes consists of a set of rules. Each rule has a condition and an action. The condition filters incoming traffic to belong to a certain class, and triggers the action for that class. A QoS policy cation includes multiple parameters value to be set for the matched class; bandwidth, queue length ... etc. Such rules are expressed as:

$$R_i := C_i \Rightarrow A_i \quad (1)$$

Each rule, R_i , maps traffic matching condition, C_i , to a specific action (set of parameters). The setting of the parameters can be viewed as the behavior this traffic class will follow. We will later refer to per-hop behavior and per-domain behavior when we model the overall policy on the domain. Packets arriving are matched in-order against each rule. The first rule that the packet satisfies its condition is triggered. The behavior associated with 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:

$$C^j = \neg C_1 \wedge \neg C_2 \dots \wedge \neg C_{j-1} \wedge C_j \quad (2)$$

Policy conflicts in this model should be analyzed on two levels; the condition and the action. The condition level is responsible for partitioning the traffic into different traffic classes (EF, AF, ... etc). A packet can only be a member of a single class. Action triggered in QoS policies require all parameters along the path of a certain class to be consistent. A class that has a certain bandwidth guarantees on one link should not be starved on another link. If a class has a maximum delay requirements on one link, it should not suffer more while traversing link on the path. Those type of inconsistencies take place due to the variations in each node capabilities over the network as well as conflicting policies configured for the same class on different nodes on the path.

The filtering condition conflict need to be checked on each node individually, and independent from other nodes. The problem in QoS policies is on the action level. Different actions may take place for the same traffic class while passing different link. Previous conflict analysis techniques suggested that any configuration inconsistencies will result in system halt. We aim here to provide a more flexible way to deal with action conflicts by proposing a fuzzy measure for the conflicts.

Conflicts analysis can be performed on multiple levels of abstractions. We first resolve conflicts within each PHB for

individual nodes. The results of this process has to a conflict-free policy, as it will be used to proceed further analysis over the domain (reflecting action conflict between different nodes on the domain). This process was depicted in figure 1.

The first step towards conflicts analysis in DiffServ environment is to model each action/PHB in a way that will simplify and facilitate the analysis. 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 $B = \lceil \log(N) \rceil$, 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 rule action is a combination of all those parameters values.

In the rest of this section we propose two orthogonal classifications of conflicts according to conflicts strength and scope of the conflicting parameters. Analyzing conflicts over PHB and PDB levels (sections IV and V) will follow this proposed classification.

A. Conflict Scopes

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

1) *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.

2) *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.

B. Conflict Strength

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.

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 it *Hard* since any slight variation in any of policy parameters will trigger a conflict.

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.

Considering our bottom-up approach of evaluating the conflicts, each parameter will be analyzed for every traffic class. A conflict arises from the inconsistencies in parameter value definition for the class along the path. Here we introduce a generic measure for analyzing parameter values conflicts. For each parameter we aim to calculate an overall measure for a certain traffic class. This measure takes into account all the routers on the path of that flow.

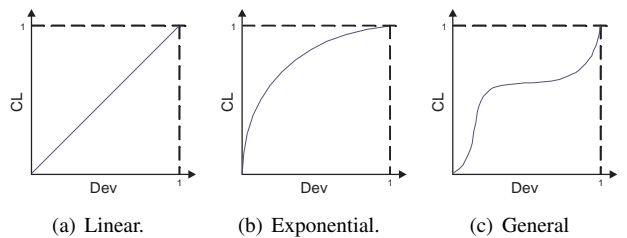


Fig. 2. Deviation-to-Conflict Level possible mappings.

For each parameter, p , on the policy action we define an aggregation method that will be used to combine different values of the parameter along the path a flow can take. The combined measure will reflect the deviation of the parameter values along the path. This measure depends on each parameter type, and the network sensitivity. For each flow f , let this measure be $Dev(p) = f(v_1, \dots, v_n)$, where v_1, \dots, v_n denote the values defined on the policy configured for devices $1, \dots, n$ for a parameter p^f . Now we can define a generic conflict measure as a function that maps the variability measure to a certain conflict level in the range $CL \in [0, 1]$.

$$CL : Dev(p^f) \mapsto [0, 1] \quad (3)$$

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. Values in-between will depend on the specific conflict description and the parameter involved (calculation of $Dev(p)$). The CL mapping 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 mapping from the deviation measure to the conflict level should be using a monotonically increasing function, $Dev = 0 \Rightarrow CL = 0$, and $Dev = 1 \Rightarrow CL = 1$. Figure 2 shows some candidate functions to be used as guidelines for administrators. The x -axis will take different deviation values, and maps it to the conflict level value on the y -axis.

IV. PHB POLICY ANALYSIS

The encoding of actions performed at individual devices for each traffic class/flow is introduced in this section. As discussed in previous sections, we consider the analysis on the parameter level. The PHB action specified in the policy will be encoded as a set of parameter values. 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.

A. 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 [10], and the parameter classification mentioned in section III. 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 Boolean expression representing a single PHB will have the following number of bits:

$$L = |V_B| + \sum_{v_i \in V_Q} [\log_2(\max(v_i))] + \sum_{v_i \in V_R} [\log_2(\max(v_i))] \quad (4)$$

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:

$$\begin{aligned} PHB_i^j \equiv & PHB_{b_1}^j | PHB_{b_2}^j | \dots | PHB_{b_n}^j | \\ & PHB_{q_1}^j | PHB_{q_2}^j | \dots | PHB_{q_m}^j | \\ & PHB_{r_1}^j | PHB_{r_2}^j | \dots | PHB_{r_k}^j \end{aligned} \quad (5)$$

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, since each parameter will be enforced independently.

A policy rule defined for flow f will be encoded as:

$$R_f := C_f \Rightarrow PHB^f \quad (6)$$

A complete mapping of all the parameters is discussed in table I.

We present action parameters as specified in the Policy QoS Information Model, [10]. 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 I 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.

Encoding Sample: 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.

B. PHB Conflict Analysis

Actions conflicts at a single device level fall in the scope of *Intra-PHB* conflicts. As mentioned earlier, those include single and multiple parameters inconsistencies. We show here how to manipulate the given policy expression for each parameter while considering the conflict strength/severity analysis. Most of the conflicts on this level are hard conflicts, since the output of this step will be used in the overall analysis. The results of PHB analysis will be a conflict-free policy expressions for all traffic classes, and parameters.

1) *Single parameter 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. The following list summarizes potential conflicts that can be detected at the parsing of the policy. This preprocessing part only consider values validation.

- Negative parameter value
- Min-Max range violation
- Out of range value
- Unified value over different flows

Therefore, we claim that all such conflicts will be discovered in this early phase as parameters validation on the input values.

2) *Multi-parameter conflict:* Those conflicts reflect the interaction between parameters and if some dependencies exist. The following example shows how to detect such conflict.

Example: Priority-Fairness Conflict (PF): If the priority parameter is set for a certain class, the fairness bit has to be 0, since you can enforce fairness while having a priority. The conflict level measure for this case assuming x_f is the bit representing fairness, and $x_1 \dots x_r$ representing the priority value will have the form: $CL_{PF} = x_f \wedge (x_1 \vee x_2 \dots \vee x_r)$. This means that if the fairness exists and the priority has a non-zero value then $CL_{PF} = 1$. If the fairness is not set, then no conflict arises ($CL_{PF} = 0$).

V. PDB POLICY ANALYSIS

Having constructed policy actions on a *PHB* level, those need to be aggregated across the domain. PDB-encoder takes as input different conflict-free PHB policies and resolve the conflicts between actions applied to the same flow. policy encoding and conflict resolution are not disjoint in this step.

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
Congestion Control	Fairness	Boolean	Single bit
	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 I
QPIM PHB PROPERTIES MAPPING

We define aggregation operators between parameters based on the methodology for solving conflicts.

A. 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 .

$$PDB_T^j \equiv PHB_1^j \bowtie_T PHB_2^j \bowtie_T \dots \bowtie_T PHB_n^j \quad (7)$$

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

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

$$PDB^j \equiv PDB_{b_1}^j | PDB_{b_2}^j | \dots | PDB_{b_n}^j | \\ PDB_{q_1}^j | PDB_{q_2}^j | \dots | PDB_{q_m}^j | \\ PDB_{r_1}^j | PDB_{r_2}^j | \dots | PDB_{r_k}^j \quad (8)$$

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.

B. PDB Conflict Analysis

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. This analysis fall in the scope of inter-PHB conflict. A 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 in terms of conflict strength or severity. Here we propose how to evaluate the conflict level measure on different types of parameters. A conflict rises when the resulting behavior is different from the expected one, or the aggregation resulted in an empty behavior.

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 must be *TRUE*:

$$PHB_{b_l}^j \equiv PHB_{b_k}^j \quad \forall l, k \quad (9)$$

where l, k are routers within the domain.

This conflict is a hard conflict, it either exists or does not.

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 measure, CL , described in section III-B. Depending on all PHBs values for this flow, a measure of deviation between them, Dev can be calculated. This measure is used afterwards to select the $CL \in [0, 1]$. 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.

Here we will chose a monotonically non decreasing function to reflect the mapping of equation 3. So, by the increase of the deviation measure, the conflict level increases. 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.

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 deviation measure, 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.

	Drop method	Queue size	Priority	Bandwidth
P1	1	3	40	20-60
P2	2	3	50	20-60
P3	1	1	10	10-30
Dev	N/A	2/3	4/5	3/4
CL	1	0.66	0.80	0.75

TABLE II
CONFLICT LEVEL CALCULATIONS FOR THE GIVEN POLICIES.

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

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.

VI. CASE STUDY

In this lase section, a simple case study will be presented. Assuming we have a network with a single path through three devices: n_1, n_2 and n_3 . We will define here policy actions for a single traffic flow for the three nodes.

P1: Queue size = 40
Min BW = 20%
Max BW = 60%
Priority = 3
Drop method = 1

P2: Queue size = 50
Min BW = 20%
Max BW = 60%
Priority = 3
Drop method = 2

P3: Queue size = 10
Min BW = 10%
Max BW = 30%
Priority = 1
Drop method = 1

Each policy will be analyzed first for any PHB conflicts. The values validation will pass, since there are no violations. We will now show in detail how to aggregate those parameters on the domain level. Here we have one boolean parameter (drop method), two quantitative parameters (queue size and priority) and one range parameter (bandwidth). For the drop method, the binary representation will have 2 bits (assuming we have 2 drop methods). The corresponding bit will be set to value "1", if its method is active.

Table II shows the values calculated using the given policy parameters. Linear mapping was used for the quantitative and range parameters (deviation value reflects directly the conflict level). The boolean parameter conflict is performed directly where the values are not equivalent on all policies.

We will now change the policies to address the conflicts of drop method and queue size, since they have the largest values.

	Drop method	Priority	Queue size	Bandwidth
P1	1	3	30	20-60
P2	1	3	30	20-60
P3	1	1	10	10-30
Dev	N/A	2/3	1/3	3/4
CL	0	0.66	0.33	0.75

TABLE III
CONFLICT LEVEL CALCULATIONS FOR THE MODIFIED POLICIES.

P1: Queue size = 30
Min BW = 20%
Max BW = 60%
Priority = 3
Drop method = 1

P2: Queue size = 30
Min BW = 20%
Max BW = 60%
Priority = 3
Drop method = 1

P3: Queue size = 20
Min BW = 10%
Max BW = 30%
Priority = 1
Drop method = 1

Table III shows the resulting values after the modifications. The conflict value for the queue size decreased to 0.33, since the new configuration has closer queue sizes defined.

VII. 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 assess 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. Combining different management policies (QoS, security, fault detection, etc) will be easier given the generic representation. A methodology for selecting the appropriate conflict level mapping needs to be developed. This can be performed by thorough experimental studies for different configurations.

The integration with high level policy representations 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.

ACKNOWLEDGMENT

This research was supported in part by Intel IT. Any opinions, findings, conclusions or recommendations stated in this material are those of the authors and do not necessarily reflect the views of the funding sources.

REFERENCES

- [1] Ehab Al-Shaer and Hazem Hamed. Taxonomy of conflicts in network security policies. *IEEE Communications Magazine*, 44(3), March 2006.
- [2] Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.
- [3] M. Charalambides, P. Flegkas, G. Pavlou, A. Bandara, N. Dulay, E. Lupu, J. Rubio-Loyola, A. Russo, and M. 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.
- [4] M. Charalambides, P. Flegkas, G. Pavlou, A. Bandara, E. Lupu, A. Russo, N. Dulay, M. Sloman, and J. 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.
- [5] Marcial P. Fernandez, Aloysio de C.P. Pedroza, and Jose F. Rezende. Converting qos policy specification into fuzzy logic parameters, 2003.
- [6] M.P. Fernandez, A. de Castro, P. Pedroza, and J.F. de Rezende. Optimizing fuzzy controllers with genetic algorithms for qos improvement. *International Telecommunications Symposium-ITS2002*, 2002.
- [7] P. Flegkas, P. Trimintzios, and G. Pavlou. A policy-based quality of service management system for ip diffserv networks. *IEEE Netwrok*, 16(2), March/April 2002.
- [8] H. Hamed, E. Al-Shaer, and W. Marrero. Modeling and verification of ipsec and vpn security policies. In *13TH IEEE International Conference on Network Protocols (ICNP'05)*, pages 259–278, Washington, DC, USA, 2005.
- [9] E. C. Lupu and M. Sloman. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering*, 25(6):852–869, November/December 1999.
- [10] Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, and B. Moore. Policy QoS Information Model. RFC 3644 (Proposed Standard), November 2003.