A Case-Based Reasoning Approach for Automated Management in Policy-Based Networks

Nancy Samaan and Ahmed Karmouch

School of Information Technology & Engineering (SITE), University of Ottawa,
161 Louis Pasteur St. Ottawa, ON, Canada K1N-6N5
{nsamaan,karmouch}@site.uottawa.ca

Abstract. Policy-based networking technologies have been introduced as a promising solution to the problem of management of QoS-enabled networks. However, the potentials of these technologies have not been fully exploited yet. This paper proposes a novel policy-based architecture for autonomous self-adaptable network management. The proposed framework utilizes case-based reasoning (CBR) paradigms for online creation and adaptation of policies. The contribution of this work is two fold; the first is a novel guided automated derivation of network level policies from high-level business objectives. The second contribution is allowing for automated network level policy refinement to dynamically adapt the management system to changing requirements of the underlying environment while keeping with the originally imposed business objectives. We show how automated policy creation and adaptation can enhance the network services by making network components behavior more responsive and customizable to users' and applications requirements.

1 Introduction

Policies have been introduced as efficient tools for managing QoS-enabled networks. It has been widely supported by standards organizations such as the IETF and DMTF to address the needs of QoS traffic management. Policies are sets of rules that guide the behavior of network components. In current systems, policies are defined by users, administrators, or operators. Once defined, these policies are translated and stored in a policy repository. Policies are then retrieved and enforced as needed.

Despite the recent research advances in the area of policy-based network management, e.g., [1, 2, 3], existing policy frameworks are faced with various challenges. Current networking systems, characterized with increasingly growing sizes and services, are becoming extremely complex to manage. Hence, an increasing burden is put on network administrators to be able to manage such networks. Furthermore, static policy configurations built a-priori by administrators into network devices usually lack the flexibility required by wired/wireless networks environments and may not be sufficient to handle different changes in the underlying environments. On the other hand, in current systems, network

A. Sahai and F. Wu (Eds.): DSOM 2004, LNCS 3278, pp. 76–87, 2004.

reconfiguration in response to users' requests for service customization can only be performed manually by a network operator. This results in significant delays ranging from minutes to days. In summary, the traditional policy-based management approach based on *Condition-Action* notion poses a major difficulty of acquiring necessary management knowledge from administrators, while it lacks the ability to deal with unexpected faults. Further, once policies are built into the network, there is no possibility to learn from gained experience. These challenges along with current advances in hardware/software network technologies and emerging multi-services networks necessitate the existence of robust self-learning and adaptable management systems.

This paper proposes a novel approach to autonomous self-adaptable policy-based networks. The proposed work utilizes case-based reasoning (CBR) paradigms [4] for on-line selection, creation and adaptation of policies to dynamically reconfigure network components behaviors to meet immediate demands of the environment based on previously gained experiences. In general, a CBR system [4] is a system that solves current problems by adapting to or reusing the used solutions to solve past problems. It carries out the reasoning with knowledge that has been acquired by experience. This acquired experience is stored in a case-base memory and used for knowledge acquisition. The CBR systems analyze and obtain solutions through algorithms of comparison and adaptation of problems to a determined situation.

In the proposed approach, policies are presented as cases. Each case (policy) consists of policy objectives, constraints, conditions and actions. Hence, the network behavior is controlled through defining a set of applicable cases. The key idea is that the network status is maintained in terms of sets of constraints and objectives. A better network behavior can than be formed on the basis of previous experiences gained from old cases that have been applied before. The network behavior is adapted by using knowledge of the network monitored resources and users' requirements to continuously change these sets of constraints and objectives. Given these new sets, the goal is to redesign these cases such that they can operate to achieve the network desired performance. A reasoning engine uses CBR adaptation techniques, such as null adaptation, parameter substitution and transformation [4] to reach this goal.

The remainder of this paper is organized as follows. In section 2 related work and existing approaches for QoS management and policy adaptation are briefly discussed. The necessary background for case-based reasoning paradigms is introduced in Section 3. The proposed policy-based management framework is described in section 4. Finally, section 5 concludes the paper.

2 Related Work and Motivation

Existing frameworks that have been developed to support QoS management mainly fall into one of two categories [5]; reservation-based and adaptation-based Systems. Although adaptation seems to provide a more promising solution for network management, existing adaptation techniques still have certain limitations. For example, many QoS-based adaptive systems use indications of QoS failure to initiate adaptation actions. Consequently, adaptation may fail in

many cases such as in the case of a QoS failure resulting from a congested link. Moreover, these techniques usually lack an essential degree of flexibility to build upon past experiences gained from the impact of previously pursued adaptation strategies on system behavior.

Policy-based network management has been introduced as a promising solution to the problem of managing QoS-enabled networks. However, static policy configurations built a-priori into network devices lack the flexibility and may not be sufficient to handle different changes in these underlying environments. Various research trends, e.g. [6], have highlighted the notion of policy adaptation and the central role that it can play in QoS management in policy-enabled networks. This notion of policy adaptation is becoming even more crucial as the managed systems become more complicated.

In [7], Granville et al. proposed an architecture to support standard policy replacement strategies on policy-based networks. They introduced the notion of policy of policies (PoP). PoPs, acting as meta-policies, are defined to coordinate the deployment of network policies. The definition of PoP requires references to every possible policy that may be deployed besides the identification of events that can trigger a policy replacement. Although their work follows the concepts of policies automation, it puts a burden on the network administrator to define both the standard policies and the PoPs. Planning policies in the existence of PoPs is a complex task. Moreover, reaching an adequate policy replacement strategy requires a complex analysis process. The administrator still has to check which policies deployment strategies were successful and which strategies failed to achieve their goals and manually update these strategies.

In [8] a genetic algorithm based architecture for QoS control in an active service network has been introduced. Users are allowed to specify their requirements in terms of loss rate and latency and then policies are used to adapt the queue length of the network routers to accommodate these requirements. The proposed work has the advantage that it is benefits from learning for adaptation.

Agents are used in [9] to represent *active policies*. The proposed architecture has a hyper-knowledge space, which is a loosely connected set of different agent groups which function as a pluggable or dynamically expandable part of the hyper-knowledge space. Active policies, which are agents themselves, can communicate with agents in the hyper-knowledge space to implement policies and retrieve information from agents. The architecture takes advantage of intelligent agents features such as the run-time negotiation of QoS requirements. However, an active policy by itself has to be created by the administrator, and once deployed to the network it remains static through its life-cycle.

In [6] a framework for adaptive management of Differentiated Services using the Ponder language [10] has been proposed. The framework provides the administrator with the flexibility to define rules at different levels. Policy adaptation is enforced by other policies, specified in the same Ponder policy notation. A goal-based approach to policy refinement has been introduced in [11] where low level actions are selected to satisfy a high-level goal using inference and event-calculus. In contrast to existing approaches, the proposed framework takes advantage of availability of previous experience gained from previously applied policies and their behavior to make decisions concerning the creation of future policies. An-

other approach has been presented in [12] which attach a description of the system behavior, in terms of resource utilization, such as network bandwidth and response time, to each specified rule.

3 CaseBased Reasoning Paradigms

Case-Based Reasoning (CBR) is a problem solving and learning paradigm that has received considerable attention over the last few years [4, 13]. It has been successfully applied in different domains such as e–commerce [14] and automated help desks [15]. CBR relies on experiences gained from previously solved problems to solve a new problem. In CBR past experiences are referred to as cases. A case is a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner. A case is usually described by a set of attributes, also often referred to as features. Cases that are considered to be useful for future problem solving are stored in a memory-like construct called the case-base memory. In broad terms a CBR reasoning cycle consists of four basic steps; namely: case retrieval, reuse, revision and retainment. A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution based on reusing a previous case, and retaining the new experience by incorporating it into the existing case-base memory.

Compared to rule-based systems, CBR does not require causal models or a deep understanding of a domain, and therefore, it can be used in domains that are poorly defined, where information is incomplete, contradictory, or where it is difficult to get sufficient domain knowledge. Moreover, it is usually easier for experts to provide cases rather than to provide precise rules, and cases in general seem to be a rather uncomplicated and familiar problem representation scheme for domain experts. CBR can handle the incompleteness of the knowledge to which the reasoner has access by adding subsequent cases that describe situations previously unaccounted for. Furthermore, using cases helps in capturing knowledge that might be too difficult to capture in a general model, thus allowing reasoning when complete knowledge is not available. Finally, cases represent knowledge at an operational level; they explicitly state how a task was carried out or how a piece of knowledge was applied or what particular strategies for accomplishing a goal were used.

4 Case–Based Policy Management Architecture

As shown in Figure 1, the main component of the proposed architecture is the case-based policy-composer (CBPC) which is responsible for translating higher-level business policies into lower-level network policies and for continuously adapting the network behavior through the online refinement of network policies in the policy repository. The CBPC relies on two different sources of knowledge for reaching decisions concerning policies changes. It continuously receives an updated view of different business-level objectives, service-level agreements (SLAs) with customers along with the underlying network topology and constraints.

The second source of information is provided by a set of monitoring agents [16] responsible for monitoring network resources based on the monitoring policies specified by the CBPC. Once obtained, the CBPC is then responsible for analyzing these knowledge to reach decisions concerning the adaptation and creation of different sets of policies; namely: admission, provisioning, routing and monitoring policies, based on previously gained experiences. The main focus of the work presented in this paper is the automated generation and refinement of admission and provisioning policies for differentiated-services operated networks [17].

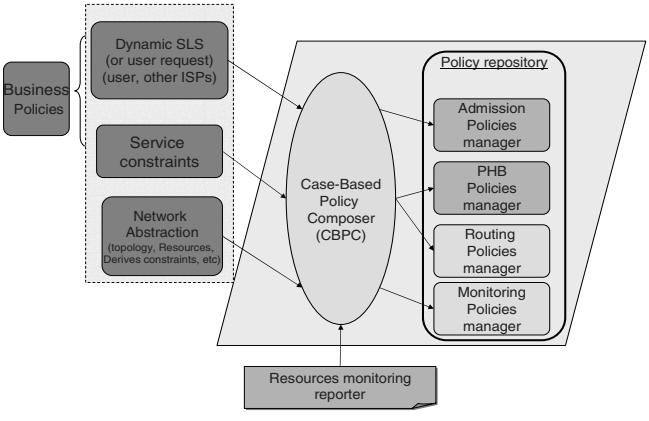

Fig. 1. Policy Management Architecture.

A detailed description of the functionalities of the CBPC is shown in Figure 2. The key idea in the proposed work is that policies are presented as cases. Hence, the terms *case* and *policy* will be used interchangeably throughout the rest of the paper. Each case (policy) consists of problem domain, describing the case's constraints and objectives, and a solution domain, describing the actions that must be taken under and which conditions to reach the specified objectives. A new policy generation/adaptation is triggered as a result of either changes in the supplied business and SLAs requirements or through objective violation indicated by information obtained from monitoring agents. The CBPC starts by deriving target objectives and constraints to represent the problem of new case. In the second step, the *retrieval step*, the CBPC uses a *similarity measure* to find previously existing cases in the policy repository with objectives and constraints that best match the target ones. Using a set of *adaptation operators*, the solutions of these retrieved cases are adapted, in the third step, to form the solution of the new target case. Once assembled, the new case (policy) is dispatched at the network level. A *refinement step* is carried-out to evaluate the behavior of dispatched policy. The case is repeatedly refined and dispatched until the target objectives are met. Finally, the new case is *retained* for future use in the case-base memory. The following sections provide a detailed description of these steps to illustrate the life cycle of policies creation and adaptation.

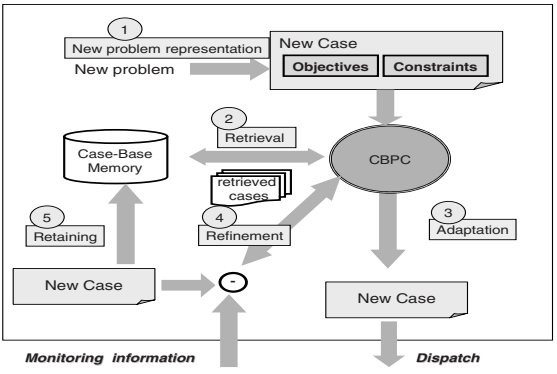

Fig. 2. Functionalities of the CBPC.

4.1 Step 1: Policy Representation and Construction

In policy-based management systems, one starts with a business-level specification of some objective (e.g., users from department A get Gold services) and ends up with a network-level mechanism that is the actual implementation of this objective (e.g., a classifier configuration for admission control and a queue configuration for per-hop-behavior treatment). The general structure of the CBPC reflects and maintains this relation between the specification of objectives and the final mechanisms passing via network-level policies through a four-level hierarchical representation of cases as shown in Figure 3. The solution of a layer i is mapped as the objectives of new subcases in the lower layer $i+1$. For example, at the highest level, abstract cases represent different business objectives and the corresponding solution is a set of finer grain network level solutions. Each of these solutions is considered an objective for a lower level case and so forth.

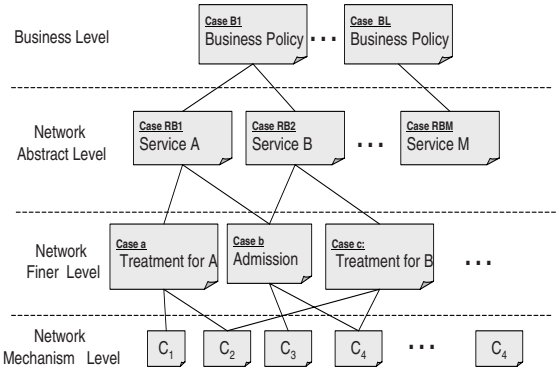

Fig. 3. General Case Hierarchy.

When a new business objective specification is posed, one or more abstract cases are retrieved and their solution is adapted to fit the specifications of the new objective. The result is a high-level description of the target solution. This high-level solution is further refined by retrieving and adapting more specific cases at each level, each solving some subproblem of the target. Eventually, concrete cases are selected and an actual set of policies can be produced. In this way, the CBPC builds up an overall solution in an iterative manner. The evolving solution forms an emergent abstraction hierarchy as high-level solution parts (from the abstract cases) are refined to produce a set of detailed policies.

Figure 4 shows a general case template, where each case Λ_i consists of a problem description part P_i and a corresponding solution S_i, i.e., $\Lambda_i = (P_i, S_i)$. Furthermore, P_i is composed of a set of objectives O_i and imposed constraints CN_i, while S_i is a set of solution steps SS_{ij}, where $SS_{ij} = (R_{ij}, C_{ij}, A_{ij}, T_{ij})$. R_{ij} is a set of roles, C_{ij} is a set of conditions, A_{ij} is the set of corresponding actions and T_{ij} is the life-time of solution step SS_{ij}.

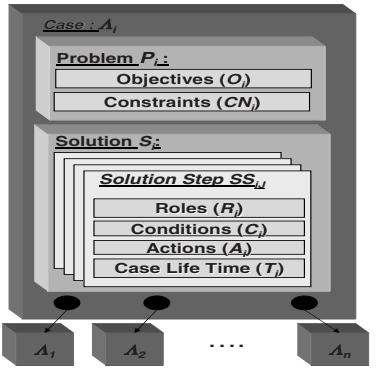

Fig. 4. Policy representation as a case.

4.2 Step 2: Policies Retrieval

During retrieval, the target objectives of the new case are matched against the described objectives of cases in the case memory and a measure of similarity is computed. The result is a ranking of cases according to their similarity to the target, allowing one or more best-matching cases to be selected. The similarity of a case Λ_i in the case-base memory to a target case $\overline{\Lambda}$ is calculated through the similarity measure defined as

$$sim(\overline{\Lambda}, \Lambda_i) = \frac{1}{\sqrt{\sum_j w_j + \sum_k cw_k}} \sqrt{\sum_j f_c(\overline{o_j}, o_{ij}) w_j + \sum_k f_c(\overline{cn_k}, cn_{ik}) cw_k} \tag{1}$$

where for each objective $o_{ij} \in O_i$, w_j is a numeric weight representing the importance of o_{ij}. Similarly, for each constraint $cn_{ik} \in CN_i$, cw_k is a numeric weight representing the influence of cn_{ik}. $f_c(x_i, y_j)$ is a local measure of similarity, defined as follows,

$$f_c(x_i, y_j) = \begin{cases} 0 & \text{if } x_i \text{ is symbolic and } x_i \neq y_i \\ 1 & \text{if } x_i \text{ is symbolic and } x_i = y_i \\ \frac{|x_i - y_i|}{range_i} & \text{if } x_i \text{ and } y_i \text{ are numeric} \end{cases} \quad (2)$$

where $range_i$ is the allowable range for x_i and y_i, used to normalize the similarity distance between the two features.

The number of retrieved cases depends on a preselected similarity threshold θ, such that a similar case Λ_i is retrieved *iff* $sim(\overline{\Lambda}, \Lambda_i) \geq \theta$.

4.3 Step 3: Policy Adaptation

Each of the retrieved cases in the previous stage undergoes a sequence of adaptation steps to meet the objectives of the target case. This stage can be referred to as *partial adaptation*. During this stage after applying a set of adaptation operators, some of the candidate cases are gradually eliminated if they failed to meet any of the target objectives. The remaining cases are then fed into the second stage for an *overall adaptation* to come up with a unified solution for the target case. Partial and overall adaptation steps are described next.

Partial adaptation. Different operators are used to adapt each of the retrieved cases separately. In the following, each of these operators is described.

- **A1: Null adaptation.** This is the simplest type of adaptation, which involves directly applying the solution from the most similar retrieved case to the target case. Null adaptation occurs when an exact match is found or when the match is not exact but the differences between the input and the target cases are known by the CBPC to be insignificant and, therefore, can be directly changed. A simple example to illustrate such a situation occurs in replacing an IP address in a classification policy, or a users' domain in a business-level policy. Figure 5 shows an example of a case adaptation using null adaptation.
- **A2: Parameter adjustment adaptation.** A structural adaptation technique that compares specified parameters of the retrieved cases and target case to modify the solution in an appropriate direction based on the parameter values in all retrieved cases. In this operator, the administrator defines a set of formulae or configuration methods according to the nature of each parameter. Figure 6 gives an example of a congestion policy adaptation using parameter adjustment operations based on two retrieved cases.
  In general, most parameters adjustments can be obtained as the average value of all recommended values from all retrieved cases as follows

$$\overline{P}_i = \frac{1}{k} \sum_{j=1}^{j=k} (P_{ji} \frac{\overline{o_l}}{o_{jl}}) \quad (3)$$

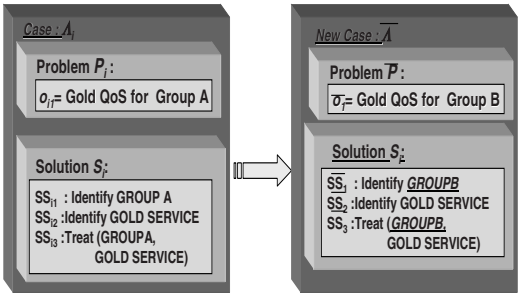

Fig. 5. An example of null adaptation.

where $\overline{P}_i$ is the new parameter value in the i^{th} solution step $\overline{SS}_i$, in the target case. k is the number of retrieved cases, and $\overline{o_l}$ and o_{jl} are the values of related objectives in the target case, $\overline{\Lambda}$, and the retrieved case, Λ_j, respectively.

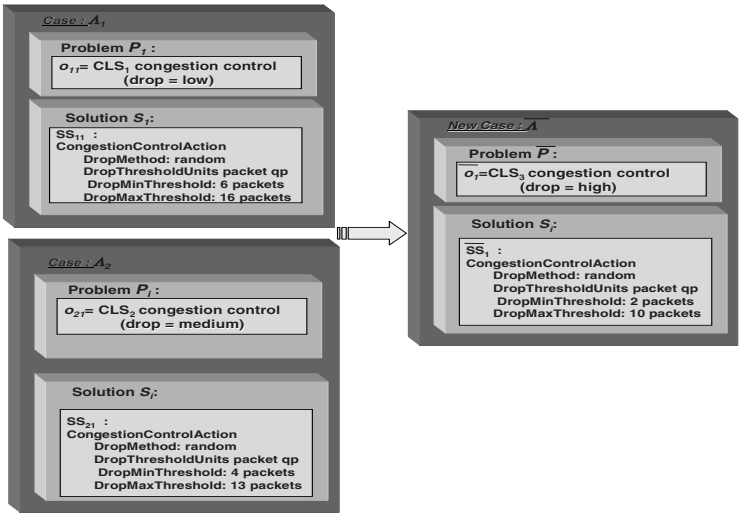

Fig. 6. Example of case adaptation using parameter Adjustment.

– **A3: Adaptation by reinstantiation.** This type of adaptation is selected when the old and new problems are structurally similar, but differ in either one or more of their constraints. In this case, reinstantiation involves replacing one or more of the old actions with a new action that is used to instantiate features of an old solution with new features.
For example, if the retrieved and target cases differ in a constraint concerning the availability of applicable mechanisms at the lowest level of the hierarchy, then the mechanism in the old solution is replaced with an equivalent mechanism in the target case that implements the same objective.

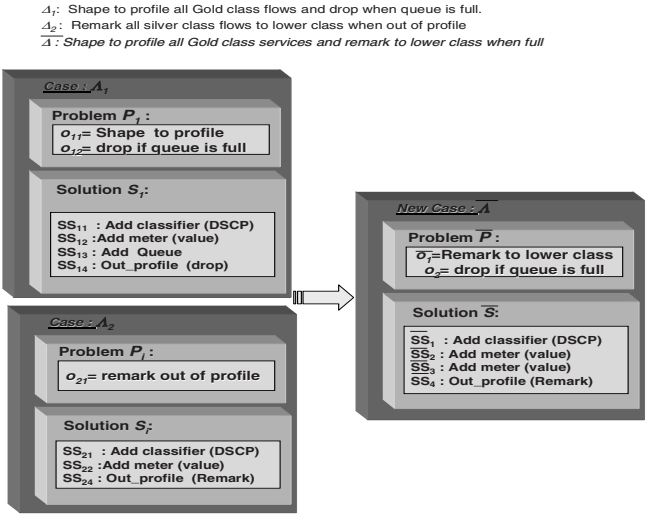

Fig. 7. A simplified example of an overall adaptation.

– **A4: Adaptation by heuristics.** This adaptation involves the utilization of a set of predefined rules set by the administrator for the purpose of case adaptation. For example, the adaptation of an admission control case can be based on the heuristic that a behavior aggregate classifier (BA) is used at edge routers connecting to other domains while a multiple field (MF) classifier is used for edge routers connected to users' hosts. Another example of a heuristic rule is that each objective in a case that includes a bandwidth allocation implies that a classifier and a queue should be allocated to the traffic class defined by the objective's conditions. Hence, once a bandwidth allocation policy is specified as a case objective, at least one classification and one scheduling action must exist as solution steps for this case.

Overall adaptation. In the case where none of the retrieved cases met the objectives of the target case after the application of one or more partial adaptation operations, an overall adaptation is performed using two or more partially adapted cases to generate the target solution. Figure 7 shows a simplified operation of an overall adaptation in response to changes in a dynamic SLA. In the Figure, two retrieved SLA policies were used to generate the required policies of a new SLA based on an overall adaptation of these two cases.

4.4 Step 4: Policy Refinement

When a new case is produced and dispatched, it has to go through a repeated cycle of evaluation and refinement before it can be finally stored in the case-base memory. As shown in Figure 8, at each refinement step i, the difference between the case's original objectives and QoS measurements obtained from the

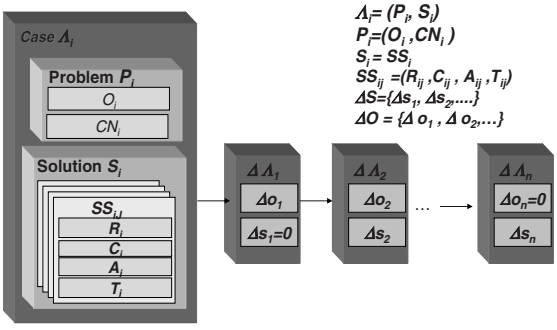

Fig. 8. Case Refinement.

monitoring agents, ΔO_i, of the leaf cases is calculated and used to perform a solution refinement ΔS_i through one or more parameter adjustment operations, described above. This cycle can be repeated several times until either the case objectives are met, i.e., $\Delta O_i = 0$, or the CBPC fails to perform any further adaptation. If the refinement is successful, the next step, case retainment, is carried out. Otherwise, if the refinement failed at the lower-level cases, it propagates to the next higher-level.

4.5 Step 5: Policy Retainment

The final step in the case life cycle is learning. Newly solved problems are learnt by packaging their objectives and solutions together as new cases, and adding them to the case memory. There are a number of issues associated with this type of learning, mainly the increasing size of the memory. However, often it is not necessary to learn an entire new case if only a small part of its solution or objectives is novel. Significant redundancy is eliminated by benefiting from the hierarchical representation of cases. Since learning can operate at a finer level of granularity as parts of different policies can be treated as separate cases.

5 Conclusions

In this paper we presented a novel framework for autonomous self- learning and adaptive policy-based network management. The proposed work utilized case-based reasoning paradigms for on-line selection, creation and adaptation of policies. The framework base policy creation and adaptation decisions on previously gained experiences of the management history. The main advantage of the proposed work is that it creates a dynamic environment where the network components are self-adaptable in response to changes in business and users' objectives. In addition, it frees up specialized administrators to other design and development tasks. In future work, we plan to evaluate our work through the implementation of the proposed architecture.

References

1. G. Valérie, D. Sandrine, K. Brigitte, D. Gladys and H. Eric, "Policy-Based Quality of Service and Security Management for Multimedia Services on IP networks in the RTIPA project", in *MMNS 2002, Santa Barbara, USA*, Oct. 2002.
2. P. Flegkas, P. Trimintzios and G. Pavlou, "A Policy-Based Qualifty of Service Management System for IP DiffServ Networks", *IEEE Network, Special Issue on Policy-Based Networking*, pp. 50–56, Mar./Apr. 2002.
3. L. Lymberopoulos, E. Lupu and M. Sloman, "QoS Policy Specification - A Mapping from Ponder to the IETF Policy Information Model", in *3rd Mexican Intl Conf in Computing Science (ENC01)*, Sept. 2001.
4. Janet Kolodner, *Case-based reasoning*, Morgan Kaufmann Publishers Inc., 1993.
5. C. Aurrecoechea, T. Campbell, A. and L. Hauw, "A Survey of QoS Architectures", *ACM/Springer Verlag Multimedia Systems Journal , Special Issue on QoS Architecture*, vol. 6, n. 3, pp. 138–151, May. 1998.
6. L. Lymberopoulos, E. Lupu and M. Sloman, "An Adaptive Policy Based Management Framework for Differentiated Services Networks", in *IEEE 3rd Intl Wrkshp on Policies for Distributed Systems and Networks (POLICY'02), Monterey, California*, pp. 147–158, Jun. 2002.
7. Z. Granville, L., A. Faraco de Sá Coelho, G., M. Almeida and L. Tarouco, "An Architecture for Automated Replacement of QoS Policies", in *7th IEEE Symp. on Comput. and Comm. (ISCC'02), Italy*, Jul. 2002.
8. I. Marshall and C. Roadknight, "Provision of Quality of Service for Active Services", *Computer Networks*, vol. 36, n. 1, pp. 75–85, Jun. 2001.
9. T. Hamada, P. Czezowski and T. Chujo, "Policy-based Management for Enterprise and Carrier IP Networking", *FUJITSU Sc and Tech Jrnl*, vol. 36, n. 2, pp. 128–139, Dec. 2000.
10. N. Damianou, E. Dulay and M. Sloman, "The Ponder Policy Specification Language", in *IEEE 2nd Intl Wrkshp on Policies for Distributed Systems and Networks (POLICY'01), Bristol, UK*, pp. 18–39, Jan. 2001.
11. A. Bandara, E. Lupu, J. Moffet and A. Russo, "A Goal-based Approach to Policy Refinement", in *Policy 2004), New York, USA*, Jun. 2004.
12. S. Uttamchandaniand, C. Talcott and D. Pease, "Eos: An Approach of Using Behavior Implications for Policy-based Self-management", in *14th IFIP/IEEE Intl Wrkshp on Distributed Systems: Operations and Management, DSOM 2003, Heidelberg, Germany, October 20-22*, pp. 16–27, 2003.
13. A. Aamodt and E. Plaza, "Case-based reasoning:Foundational issues, methodological variations and system approaches", *AI Commu.*, vol. 7, pp. 39–59, 1994.
14. R. Bergmann and P. Cunningham, "Acquiring Customers' Requirements in Electronic Commerce", *Artif. Intell. Rev.*, vol. 18, n. 3-4, pp. 163–193, 2002.
15. M. Goker and T. RothBerghofer, "Development and Utilization of a Case-Based Help-Desk Support System in a Corporate Environment", in *3rd Intl Conf on Case-Based Reasoning, ICCBR-99, Seeon Monastery, Germany, July*, 1999.
16. N. Samaan and K. Karmouch, "An Evidence-Based Mobility Prediction Agent Architecture", in *Mobile Agents for Telecommunication Applications, 5th Intl Wrkshp, MATA 2003, Marakech, Morocco*, Oct. 2003.
17. S. Blake et al., "AN Architecture for Differentiated Services", IETF RFC 2475, Dec 1998.