A Categorization of Policy Conflicts in Distributed Systems and Network Management

Sarandis Mitropoulos*
Ionian University, Greece
Filosofon & Tzeveleki St., 31100 Lefkada

Email: smitropoulos@ionio.gr

Antonios Gouglidis and Vasileios Giotsas
School of Computing and Communications
Lancaster University, Lancaster
United Kingdom, LA1 4WA

Email: a.gouglidis@lancaster.ac.uk
Email: v.giotsas@lancaster.ac.uk

*Corresponding author

Abstract
Policy conflict detection in distributed systems and network management is a crucial issue that strongly influences the management efficiency. In this paper, after investigating thoroughly the related approaches on management policy conflict categorization, a new systematic categorization is proposed that is based on four main perspectives, namely, modality, manageability, interoperability and system specification. Subsequently, an organizational network structure that is managed through policies depicts the high applicability of our approach in policy conflict detection, while implementation examples are also given. Finally, conclusions and future work are discussed.

Keywords - Management Policy, Policy Hierarchy Analysis, Policy Conflicts, Open Distributed Processing, Model checking, PlusCal.

Paper type – Research paper

Biographical notes: Sarandis Mitropoulos is an Associate Professor of Management Information Systems in the Regional Development department of Ionian University. He also teaches and conducts senior research in the Department of Informatics of the University of Piraeus since 2004. From 2002 to 2018 he was a special scientist in informatics (system analyst) in a bank supervised by the Ministry of Finance. He received his Diploma and PhD as an Electrical and Computer Engineer from the National Technical University of Athens. He holds an MBA degree for executives from the Athens University of Economics and Business. He has also extended experience in the private sector working in ICT Research and/or Development projects.

Antonios Gouglidis is a Lecturer (Assistant Professor) at the School of Computing and Communications at Lancaster University, a member of the Systems Security Group (SSG) and the research area lead of Distributed Systems at Security Lancaster. His research

interests include the design of secure models and systems. He currently investigates access control related topics, Cloud security and the application of model checking techniques to formally verify security related properties. His research work has been published in over 40 peer-reviewed journals and conferences. He participated in several projects funded by the European Commission (FP7 and Horizon 2020) and the UK Government. Previously, he worked in academia as a Security Researcher, in the industry as a Software Engineer and in the public sector as an IT Training Consultant.

Vasileios Giotsas is a lecturer at Lancaster University, where he leads the Networks research area of the Lancaster Security Institute. He received his PhD from University College London (UCL). His research focuses on network measurements and the analysis of the routing system. He worked as a research scientist at the Internet Measurement and Analysis group of Technischen Universität Berlin, and as postdoctoral researcher at CAIDA (Center of Advanced Internet Data Analysis) at the University of California San Diego.

1 Introduction

Modern computer networks are characterized by increasingly large size and complexity, significant heterogeneity of the participating devices, transient set of users and diversity of the supported applications. As a result, a manual configuration and administration of such networks is virtually impossible. Moreover, the fact that computer networks are tightly coupled with the business goals of the organizations that deploy corporate networks mean that network administrators should also be specialized in business processes. Such experts are very uncommon and inflict great manpower cost.

Autonomic network management has emerged as a solution to the problem of efficient network management by automating the decisions of network governance and distributing the decision-making tasks to the network devices that need to take an action. In that model, the human intervention occurs at a higher level of network management. Network administrators should provide a set of policies that define the behaviour of the network and of its users under specific conditions. These policies should reflect the business goals and strategies, as well as the desired response to unexpected incidents resulted from malicious users, hardware failures or system constraints.

Due to the importance of policy-based network management (PBNM), working groups of Internet Engineering Task Force (IETF) standardized the PBNM environment (IETF, 2005) and proposed a terminology (Aiken et al, 1999) as well as a Quality of Service/QoS Management Framework (Al-Jawad et al, 2018) for various aspects of PBNM like software-defined networks. However, several different proposals describe architectures for policy-based administration. According to the IETF draft, a policy is a set of rules that define the execution of a set of actions when a set of conditions

is fulfilled. The rules are supplied by a central policy repository and are transmitted to network devices (Policy Enforcement Points–PEPs) through the Policy Decision Point (PDP) that translates the policy to an understandable format.

The remainder of this paper is organized as follows. In Section 2, we set the policy conflict problem and in Section 3 we elaborate on related systems and work. In Section 4, we propose an integrated approach in policy conflict categorization useful for detecting policy conflicts in large-scale distributed systems. In Section 5, we revisit them through a set of policy examples in an organizational intranet/extranet and demonstrate how these can be specified in PlusCal and verified using model checking. Lastly, Section 6 elaborates our conclusions and future work.

2 The problem space

Although the use of network policies greatly simplifies network management, the complexity of a network may still require non-trivial management (Katsikogiannis et al, 2018; Kallergis et al, 2020). As the network increases in size and heterogeneity, the number of policy rules that should be supplied increases, while there can be many different policy authors (Katsikogiannis et al, 2013; Phan et al, 2008; Raymer, 2006). Some of these policies inevitably apply to the same PEP or concern the same traffic. If the definition of polices is not carefully structured, it is probable that two or more policies will conflict. An example of conflicting policies is when a policy allows the traffic from an IP address while another one blocks the traffic from the same address. This is a very serious issue in PBNM, since conflicts between policies obviously lead to non-desirable and unpredictable network behaviour. As the size and structure complexity of a management system increase, more complicated relationships take place, increasing consequently the possibility of conflicting policies due to bad situations. It is necessary these conflicting policies to be detected before their enforcement and for undesirable situations to be avoided. Towards this direction, to ensure that a set of policy rules is conflict-free, a policy management tool must validate the defined policies, detect existent or potential conflicts, and resolve them by executing the most appropriate action.

Although policy validation has been a very active research topic, it is still challenging to standardize a methodology for consistency checking and resolution, mainly because policy conflicts can be very diverse in their nature (Moffett and Sloman, 1994; Charalambides et al, 2009). This diversity of conflicts demands adaptive algorithms for detecting and resolving them according to their category. Then, a unified solution should

incorporate and manage the output of these algorithms. The above statement illustrates the need for a comprehensive policy classification that provides the proper distinction between the various characteristics of conflicts in a way that will facilitate their detection and resolution. Again, this classification is non-trivial as the huge variety of applications - from Service Level Agreements to power management in Wireless Sensor Networks - renders the definition of a global conflict taxonomy to be a cumbersome process. As a result, several different conflict categorizations have been proposed by researchers, each of which pertains to a specific dimension of policy rules enforcement. The existent categorizations are either disjoint or very generic in a way that does not facilitate adequately detection and resolution. For this reason, in Section 5, we present a set of policies that cannot be categorized wholly or just partly according to existent schemes. On the other hand, our proposed scheme aims to bridge the gaps and offer a new perspective in the research of policy conflict categorization. Following this rationale, we explain in detail how our scheme completes the existing literature.

Thus, our first goal is to present and analyse the proposed categorizations thoroughly and illustrate how they facilitate the detection of conflicts. We extended existing work and propose a *new integrated categorization* that covers the widest spectrum of policy conflicts and provides detailed description of the particularities of each category and how these can be utilized in conflict detection research. Our approach covers all the aspects of a distributed system lifecycle, namely, modality, manageability, interoperability and system specification.

3 Related Work

Moffet and Sloman present in (Moffett and Sloman, 1994) a classification of policy conflicts based on the types of overlaps between imperative or authorization policies. According to this classification there are two main categories: *modality conflicts* and *goal conflicts*. Modality conflicts occur when the subject, the action, and the target of two different policies overlap, but the modalities differ. Namely, one policy permits or obliges the subject to execute an action, while the other policy blocks the subject from executing the same action (Kuhlisch, 2017). Modality conflicts can be detected simply by a syntactical scan of the policies and no application knowledge is required. On the other hand, goal conflicts are application-specific and are divided as follows:

• *Conflicts for resources*: this type of conflicts arises when two policies authorize or oblige the use of the same resource, but the available amount of this resource is not enough to serve both policies.

• *Conflicts of duties*: when a subject is assigned two contradictory duties.

• *Conflicts of interest*: when a subject is authorized to perform tasks on two different objects, the task performed on the first object may conflict with the interest of the second object.

• *Multiple manager conflicts*: multiple manager conflicts may arise when different managers manage the same target. This is problematic if their actions are executed concurrently and affect the same target aspect.

• *Self-management conflicts*: depending on the application context, a policy subject should not be able to manage itself. Such a conflict is more obvious when the subject and the object of a single policy overlap each other.

Lupu and Sloman extend in (Lupu and Sloman, 1999) the above categorization by grouping modality conflicts into three distinct categories, based on the type of the policy (authorization or obligation) and the modality:

(i) *Authorization conflicts (A+/A-)*: a policy authorizes a subject to perform some action while a different policy prohibits the same subject from performing this action.

(ii) *Obligation conflicts (O+/O-)*: a policy obliges a subject to perform some action while a different policy obliges the same subject not to perform this action.

(iii) *Unauthorized obligation (O+/A-)*: a policy obliges a subject to perform an action while a different policy prohibits the same subject from performing this action. It is also mentioned that the scenario when a subject is authorized to perform an action but required by an obligation policy to refrain from performing this action (A+/O-) is not a conflict.

Jajodia et al. (1997) follows a similar approach, identifying that conflicts can be either static or dynamic. *Static conflicts* can be detected by simply scanning the syntax of a policy statement, because their occurrence is independent of the application state and semantics. *Dynamic conflicts* arise because an application state makes two different policies incompatible. The detection of such conflicts requires a thorough knowledge of all the application states and can be made with many methods such as Linear Regression Attack with F-Test (Foresti and Persiano, 2016).

In (Chadha and Kant, 2008) the authors extend further the classification of (Lupu and Sloman, 1999; Moffet and Sloman, 1994) proposing the classification illustrated in Figure 1.

Figure 1: A classification of policy conflicts

Application-specific conflicts are divided into three categories: mutex conflicts, redundancy conflicts and other conflicts. The last category contains the conflicts described by Moffett and Sloman (1994). *Mutex (Mutually Exclusive Configuration)* conflicts occur when two different policies try to apply different actions to the same target, setting the same target parameter to two different values. Redundancy conflict exists when two different policies are identical.

Specifically, redundancy conflicts can be application dependent when their detection demands knowledge of application semantics. A typical example is when two policies define different actions that may have the same result. For instance, if policy A1 allows traffic from the National Technical University of Athens (NTUA), and policy A2 allows traffic from all the academic institutions in Greece, then, policy A1 is *redundant*, but to detect it, we need to know that NTUA is an academic institution in Greece. Another example is when policy B1 allows Voice-Over-IP calls until 1:30 pm and policy B2 allows VOIP calls until lunch hour. Depending on when the lunch hour takes place, one of the two policies is redundant. If lunch hour is after 1:30 pm, policy B2 is redundant. Otherwise, the redundant policy is B1.

Mutual exclusion (Mutex) conflicts again depend on the application semantics whether two actions are mutually exclusive or not. For example, policy C1 stops the execution of a program when intrusion is detected, while policy C2 begins the execution of the same program when a data process is requested. Normally, these two actions are not mutually exclusive unless they happen simultaneously.

In (Stone et al, 2001; Paoli et al, 2017) policies are grouped in two different categories: intra-policy and inter-policy conflicts. *Intra-policy conflicts* are caused when the conditions of at least two policies are simultaneously satisfied, but the execution of the actions of these policies is not feasible at the same time. *Inter-policy conflicts* are described as two or more policies that, when applied to the network, result in conflicting configuration commands, specified for one or more network devices. In this case, the conflict exists when the policy is applied to a specific network or device(s).

Dunlop et al. (2003) classify conflicts into four broad categories assuming an inconsistency with the assigned subject roles:

• *Internal Policy Conflicts* may occur when the tasks of a role are incompatible with each other.

• *External Policy Conflicts* occurs when different roles are incompatible only when assigned to the same user due to the existence of conflicting policies

• *Policy Space Conflict* occurs when two or more policy spaces manage the same set of subjects and attempt to enforce different and conflicting policies over them and, finally

• *Role Conflict* occurs when a user obtains a set of incompatible role assignments.

A broad work on conflicts classification has been done for security policies. Classifications of distributed firewall policy conflicts are presented in (Al-Shaer and Hamed, 2003; Haded and Al-Shaer, 2006; Pisharody, 2017). Within a single firewall the defined policies may include *intra-firewall anomalies,* where the filtered packet may be managed by more than one policy. In distributed firewall environments, the defined policies may include *inter-firewall anomalies*. In this case, the same traffic is filtered by different conflicting policies. Both the intra-firewall and the inter-firewall anomalies are further categorized (Valenza, 2017). The different intra-firewalls anomalies are described below:

(a) *Shadowing anomaly*: This type of anomaly occurs when a policy is never activated since previous policies manage all the packets that the shadowed policy would manage.

(b) *Correlation anomaly*: A correlation anomaly occurs when two different policies are partly shadowed between each other.

(c) *Generalization anomaly*: If a policy covers all the packets that a preceding more specialized policy manages, then the second policy is a generalization of the more specialized policy.

(d) *Redundancy anomaly*: A policy may be redundant if it performs the same action on the same packets as another rule. In this case, the removal of the redundant policy will not affect the firewall operation.

(e) *Irrelevance anomaly*: A policy in a firewall is irrelevant if its statement cannot match any traffic that flows through this firewall. Consequently, this policy does not affect the firewall operation and can be safely removed.

Inter-firewall anomalies have similar categories as intra-firewall anomalies, including shadowing anomalies, redundancy anomalies, correlation anomalies and spuriousness anomalies. The last category is only different in inter-firewall conflicts. A *spuriousness anomaly* occurs if an upstream and a downstream firewall manage the same traffic with opposite rules, as for example the upstream firewall permits the traffic that the downstream blocks.

The literature also contains categorizations for more specialized cases. For instance, the authors of (Syukur and al, 2005) present a classification of policy conflicts in pervasive computing environments. According to this classification, the first category of conflicts is the *policy space modality*

conflict. They occur as the space assigns different specifications on entity tasks. These differences lead to a potential or actual conflict. A *potential conflict* has not happened yet at the time of the detection of the conflict scenario, as the conflict conditions have not been met. The potential conflict is further classified into two sub-categories: *possible potential conflict* and *definite potential conflict*. The occurrence of a possible potential conflict is less possible than the definite potential conflict. This conflict can be avoided even if its occurrence conditions are met. On the contrary, a definite potential conflict will occur if the user is in the right context. The second broad category of conflicts in pervasive environments is the *role conflict* that occurs because of the different privileges that an entity possesses. A user with higher role level – hence, higher privileges - may override the execution of a shared service that has been started earlier by a user with lower role level. This results a conflict situation. The last category of conflicts described in (Syukur and al, 2005) is the *entities conflict*. Such conflicts occur when the actions that the users are specified to perform on a shared service are conflicting with each other.

Charalambides et al. (2005) as well as Alquhayz et al. (2019) have identified several potential conflicts related to obligation policies of PONDER and PONDER2, respectively. The first category is *redundancy conflicts* that arise because of duplicate policies (overlap of subjects, targets, actions and action parameters). *Mutual Exclusion (ME) conflicts* are the second domain independent category that happen when two or more directive specifications of the same network aspect are managed. *Bandwidth allocation conflicts* constitute the first category of application specific conflicts and are related to how a link capacity is assigned. Moreover, *Routing conflicts* concern how routes are assigned to meet the QoS specifications.

Finally, the authors of (Reiff-Marganiec and Turner, 2004) identify five dimensions of policy conflicts: policy types, domain entities, roles, policy relations and modalities (Table 1). The first dimension connects policy conflicts with the different *types of policies* that can be enforced (Unconditional and Conditional Goals, Unconditional and Conditional Event-Condition-Action policies). The second dimension of conflict categorization is related to *domain entities***,** which are either individual users or roles. In this case, conflicts usually arise when the entities belong to the same domain since different domain entities usually interact. However, entities that belong to different branches of the same domain may conflict if these entities have a specific relation (e.g., when the one entity is a user, and the other is a role). The third dimension classifies conflicts with respect to the users' *roles*. Such conflicts arise when users hold more than one role

or when conflicting policies are defined both for roles and individual users that have at least one of these roles. The dimension of *policy relation* characterizes conflicts according to the relation kind between the conflicting policies. A refinement conflict occurs when a policy is derived from another policy, specifying a similar but more specialized action, condition, or event. The opposite is a relaxation policy. The last dimension is *modality* conflicts where two more sub-categories of conflicts are added to the authorization and obligation modalities identified by Lupu and Sloman (1999) as well as Moffett and Sloman (1994). These are temporal modalities that refer to the execution time of an action and preference modality conflicts that are derived from conflicting goals of different network users.

Table 1: Dimensions of policy conflicts as identified by Reiff-Marganiec and Turner in (Reiff-Marganiec and Turner, 2004)

In addition, it is worth mentioning that deadlock conflicts have been detected in multithreaded applications, based on source C code conflict detection analysis (Giebas and Wojszczyk, 2020).

Finally, several large-scale outages have taken down large chunks of the Internet due to policy conflict issues. An example is the service disruption that occurred in the Seoul Region in 2018, where EC2 instances experienced DNS resolution issues. This was caused by a reduction in the number of healthy hosts that were part of the EC2 DNS resolver fleet, which provides a recursive DNS service to EC2 instances. The service was restored when the number of healthy hosts was restored to previous levels[1].

Another example is when Google Compute Engine (GCE) instances and Cloud VPN tunnels in europe-west4 experienced loss of connectivity to the Internet. The incident affected all new or recently live migrated GCE instances. The issue was caused by an unintended side effect of a configuration change made to jobs that are critical in coordinating the availability of load balancers, named Maglevs. The change unintentionally lowered the priority of these jobs in europe-west4. The issue was subsequently triggered when a data centre maintenance event required load shedding of low priority jobs. Then the automated monitoring alerted Google's engineering team to the event. The team discovered the root cause and started reverting the configuration change and the connectivity was immediately restored[2].

[1] https://aws.amazon.com/message/74876/
[2] https://status.cloud.google.com/incident/cloud-networking/18013

4 An Integrated Approach in Policy Conflict Categorization

Although the above approaches in management policy conflict categorization are significant, there is still a lack of an integrated policy conflict categorization framework from all the *system specification* and *operation* perspectives. This is imperative especially, in complex large-scale distributed computing systems where a huge number of policies is implemented in millions of resources. Considering that these policies must serve the contracted SLA's as well as the high-level business goals, special care must be taken to detect as many as possible policy conflicts. Otherwise, important operating problems may occur. Although the approaches, presented in related work are valid from various perspectives, they do not provide a global high-level view of all the policy conflicts that may occur in complex large-scale organizational distributed computing environments.

For such a reason, we need another integrated approach to cover all the system lifecycle perspectives. Hereafter, we present such an integrated categorization that determines four main policy conflict perspectives:

(i) the *modality* perspective.
(ii) the *management* perspective.
(iii) the *interoperability perspective.*
(iv) the *specification - design perspective*.

These main perspectives of policy conflicts are divided in sub-categories that are presented in detail below.

Modality as a term in the *modality perspective of policy conflicts* imposes the positive or negative authorizations and the positive or negative obligations. For example, conflicting authorization policies arise when there is a negative authorisation along with a positive one for the same subject (manager), the same target (managed objects) and the same actions over the same time frame. If a goal policy implies some actions without the respective policy subject being issued by the corresponding authorizations, a modality conflict also arises. Thus, in this perspective, we have a policy conflict between *contradictory authorizations* or due to an *obligation without the required authorization*. Detailed analysis on modality conflicts can be found in (Lupu and Sloman, 1999).

Regarding the *management perspective of policy conflicts***,** we first focus on the policy refinement process that has an important impact on the consistency of a management policy hierarchy. When refining policies from a high management level to a lower one, there is a possibility for inconsistencies to exist between the new existing policies. The latter may happen since the policy activity at one hierarchical level may not be compliant with the policy activity at another or the same hierarchical level.

In such a case, a *hierarchical (inconsistency) policy conflict* occurs. Policy conflicts also arise when managers (human or machinery) have to balance between different facts and estimate different operational statuses, and they *fail to develop appropriate strategies and action plans.* In such a case, managers that have intelligence capabilities can make dynamic decisions, but the plans they develop (e.g., serialized execution of several policies) are not corrected causing policy conflicts (wrong policies at the wrong time). In this perspective, another well-known issue that may cause policy conflicts is the availability of resources. Lack of resources imposes prioritization in their use. The best utilization of resources is one of the main management goals. Thus, the resources usage must be *accurately forecasted.* Furthermore, *the bad organization of the under-management system into management domains* is another well-known reason for policy conflicts. An appropriate and efficient setup of the managed system into management domains is very important. This management organization is directly related to the enforced definition of policies on it. For example, setting up an antivirus protection domain implies that antivirus policies should be enforced on it. Towards this direction, within a management domain, users, roles, permissions, obligations and processes must be well defined.

The *interoperability perspective of policy conflicts* mainly focuses on communication, inter-working or domain inter-influence problems. More specifically, policy conflicts may occur in case of *bad synchronization of actions of totally different (non-overlapping) environments.* Namely, the enforcement of a policy in an organizational environment may conflict with another policy enforced in another organizational environment, although there is not overlap between these two organizations (Moffett and Sloman, 1993; Lupu and Sloman, 1999). For example, this happens when a mobile telephony corporate policy for configuring its monitoring system demands an unexceptional negative interception authorization, which conflicts with a policy of an independent authority (e.g., a prosecutor) that asks for interception of calls in some specific situations. There are also policy conflicts that occur due to the *different nature of manager entities,* as for example, a human manager (administrator) and an automated (machinery) manager. The border of interpreting business level policies into computing system or network level policies is a usual cut-off point for bad policy interpretation and inconsistencies. In addition, the *different technology* of the management authorities may lead system policies in conflict. For example, the policies setup in a Windows 2016 server may not be compliant with the policies setup in a UNIX server. Thus, to mitigate policy conflict problems in new cloud-based environments the engineering teams of

Windows Server and Azure made possible for Linux VMs to be able for deployment under the Windows Server 2016 Hyper-V without problems (Warner, 2018).

Finally, we examine *the specification - design perspective of policy conflicts* according to ISO/ODP standard (Farooqui et al, 1995; Linington et al, 2011). The *bad specification and design of a distributed computing system* may be one of the main reasons for policy conflicts. The system specifications incorporate all the characteristics and parameters upon which several policies are enforced. Although policies may be appropriately defined, system malfunction due to bad specifications may lead to policy conflicts that are not so obvious at a first glance. In short, a bad specification should be fixed in advance since it may be more difficult to be detected at a later stage. According to the ISO/ODP standard, a distributed processing system is described from five viewpoints: enterprise, information, computational, engineering and technology. The conflicts that arise inside these viewpoints can be considered as bad specification policy conflicts. That may be because business goals are not consistent with each other; or there are bad computational processes; or the system engineering can never meet the QoS requirements; or the component technologies are not interoperable (Chen and Vernadat, 2007; Boiten et al, 2000). Furthermore, there can be conflicts between policies enforced on domains of different ODP viewpoints. Such conflicts mainly occur due to bad correspondences between domains of different ODP viewpoints. This is an innovative approach in policy conflicts.

Figure 2 depicts the proposed policy conflict categorization. Although hybrid policy conflict types are also possible, their analysis is out of scope in this paper.

Figure 2: The integrated approach in policy conflicts categorization

With the proposed model, a holistic approach is developed where we categorize policy conflicts in such a way to minimize the possibilities for errors in the operation of a distributed system or network. Based on the analysis of the relevant work described in Section 3, no other approaches have focused, to the best of our knowledge, on categorizing management policies through multiple perspectives. This categorization constitutes the main contribution of the paper.

5 Examples of Policy Conflicts in an Organizational Environment

To illustrate the proposed integrated categorization of policy conflicts, we provide some examples of an organizational network structure that is

managed through policies. The examined company offers application hosting services, including multimedia streaming servers, web servers and backup servers. Customers can use these servers to multicast asynchronously video, remote conferences, host web sites or ecommerce services or backup digital content. There are two types of customers: individual customers and corporate customers. The organization has three different management sub-domains: the *server farm*, the *IT department* and the *accounting department*. These departments are interconnected over a WAN. The accounting department has a VPN connection with a bank extranet. Also, the server farm department allows VPN connectivity to the customers enabling them to manage their content. Let us assume that the following two policies are defined:

- *P1: Accountants must backup their documents hourly*
- *P2: If available b/w is less than 1 Mbps, accounts should refrain from backup*

Policies P1 and P2 conflict between each other, as P1 obligates accountants to backup, while P2 prohibits backup when available bandwidth is low. This is a *modality conflict*.

An example of *authorization modality* conflict is given considering the following two policies:

- *P3: Customers are allowed to access their user directory in the Web Server*
- *P4: Between 14:00 – 16:00 only the IT department can access the Web Server for maintenance*

These two policies conflict since P3 authorizes customers to access the web server, while P4 defines an exception to this rule.

Let us examine an example of *management perspective* where *bad provisioning (hierarchical policy inconsistency)* leads to bad service for all requests. Consider the following policies:

- *P5: If good service for corporate customers' requests cannot be achieved for a specific session, refund half amount of billing*
- *P6: If good service for individual customers' requests cannot be achieved for a specific session, refund a fourth amount of billing*
- *P7: If corporate customers' request takes longer than 15 seconds, allocate 5% more bandwidth (b/w)*

- *P8: If individual customer's request takes longer than 25 seconds, allocate 5% more b/w*
- *P9: If more b/w is needed both from corporate and individual customers, give priority to corporate customers*

In this case, there is no obvious conflict among these four policies. However, the lower-level policies should be aligned with the higher-level policies. A usual high-level policy for commercial companies is to maximize profit or minimize loss (e.g., Px). Suppose that two flows, an individual and a corporate, do not meet their SLA and demand more b/w, but there is available b/w only for one of two. According to P9, the flow from corporate client takes priority and uses the extra b/w. However, it is possible that the extra b/w cannot help the flow to meet its requirement, but it would be enough for the individual flow to meet its requirements. According to the current approach the company must refund both clients, violating Px, but if the extra b/w was allocated to the individual flow, only the corporate client would be refunded. However, we must notify that such management conflicts must be resolved during the rules engineering.
Another *management conflict* is the following:

- *P10: When a customer requests to use the backup service with gold priority, follow the next steps:*
  1. *Allocate disk space and b/w*
  2. *End request to accounting department:*
     a. *If successful accept data*
     b. *If unsuccessful de-allocate b/w and disk space and send error message*

P10 defines that upon a request for backup service, the requested resources are reserved before checking the financial request status. If the order accounting fails, then the reserved resources are wasted. This is in direct conflict with the goal of making *optimal use of the available resources.* Towards to the same direction as the previous example, we must notify that this type of management conflicts must be avoided from the beginning by correctly designing the system, as for example by checking the success of purchase first, before the allocation of any resource.

A *last type of management policy conflicts* pertains to the well-known *domain overlap*. Management domain conflicts are usual in complex networks as users or machines can belong to more than one logical or network domains. Assume the following policies:

- *P11: Departmental computers must restart daily at 21:00 for maintenance*
- *P12: Backup servers must be always on*

In this case, when a departmental computer has also the role of backup server, the policies P11 and P12 are conflicting. We would like also to notify that it is very essential to avoid the bad design of policies from the initial phase.

Interoperability conflicts can arise from using different technologies. A simple scenario is when different low-level technologies are used to fulfill the same high-level goal. For instance, to allow a customer to access a service two simple authentication types are the IP address filtering and SSL:

- *P13: If the IP address is trusted, allow access to the Stream server*
- *P14: Allow access to the Web server through SSL handshake*
- *P15: If a customer logged in the Stream server, she can access the Web server*

As we can see from policies P13-P15, the Stream Web servers authorize the same customers with different methods. On the other hand, they trust each other users. This may lead to a policy conflict since a customer can use the Web server through IP address filtering and not SSL handshake, which is required by P14.

Assume the ODP-based domains of the examined network as in Figure 2 and the following policies:

- *P16: The User Agents (UA) that corresponds to the customer domain D1, have a gold SLA and must retrieve video streaming in high rates*
- *P17: The Video Streaming Servers (VSS) of the domain D2 are authorized to transmit video streaming in high rates*
- *P18: The VSS's of Domain D3 are authorized to transmit video streaming in medium rates*

According to P16-P18, D1 corresponds to D2, thus the users of D1 must be connected to the VSS's of D2. Otherwise, P16 and P18 will not be consistent (Figure 3). We would like to notify that this domain correspondence must be initialized from the initial system setup, but this condition in a dynamic environment must be checked in a continuous way, when dynamic conditions impose the Users Domain to be corresponded to another domain.

Figure 3: Domain Correspondence in ODP viewpoints

5.1 Specification and verification of example policies

In this section, we provide additional information on how to formally express some of the above-mentioned policies, and furthermore, how to verify relevant conflicts (e.g., modality, authorization) in the policies. For that, we use PlusCal to specify the policies and temporal logic properties (conflicts verification). PlusCal is a language for writing algorithms that can be translated into a TLA+ model that can be checked with the TLA+ tools. TLA+ is a high-level language for modelling programs and systems that is useful for eliminating fundamental design errors (Lamport, 2020). Due to paper space limitations, hereafter we have modelled the logic in two of the examples mentioned above, i.e., the logic of P1, P2 and of P2, P3.

Based on previous work in (Grompanopoulos, et al., 2021), we specify the *P1* and *P2* policies described above. The PlusCal specification of the policies and the modality conflict property are provided in Appendix A. In the following, we elaborate on the rational of the specification. Initially, a set of variables are defined to express the following:

- ***Hour****: An hour has passed, and a backup must be taken.*
- ***Bw_lt_1mb****: The bandwidth is less than 1 MB.*
- ***Backup_done****:A backup has been taken; and,*
- ***Role_is_accountant****: The user is an accountant.*

Each of the above variables may be of Boolean type and get values from a set BOOLEAN = {True, False}. During the initialization, we consider that a backup does not exist (Backup_done = False) and we allocate a random value from BOOLEAN to each of the remaining variables. Subsequently, we assume the following logic to implement *P1* and *P2.*

- *If an hour has passed (Hour = True) and the user is an accountant, (Role_is_accountant = True) then*
    - *If the bandwidth is 1 MB or greater (Bw_lt_1mb = False) a backup should be taken (Backup_done = True).*
    - *Otherwise, if the bandwidth is less than 1 MB (Bw_lt_1mb = True) the backup cannot be taken (Backup_done = False).*

Additionally, we define a temporal property to check the modality conflict in the aforementioned algorithm. Such a property may check that when an hour has passed, and the user is an accountant, it leads to the creation of a backup. When verifying the property using the model checker (i.e., TLC) it is shown that it gets violated and a countermeasure is provided. More

specifically, there is a system behaviour where a backup cannot be taken, even when an hour has passed (Hour = True) and the user is an accountant (Role_is_accountant = True). That happens when the bandwidth remains always below 1 MB.
Following, we further specify the policies *P3* and *P4* described above and verify the authorization modality conflict. The PlusCal specification is provided in Appendix B. Below, we elaborate on the specification rationale. We define a set of variables to express:

- ***isCustomer****: Someone is a customer and requests an access to a resource.*
- ***canCustomerAccess****: A customer is granted access on the requested resource and,*
- ***isRestrictedTime****: A variable to flag a restricted time slot.*

Each of the above variables may be of Boolean type and get a value from a set BOOLEAN = {True, False}. During the initialization, we consider that someone is a customer (isCustomer = True) and can access the resource (canCustomerAccess = True). Considering policies *P3* and *P4*, we have:

- *Customer access request may be randomly created at any time, and upon request access should be granted.*
- *During 14:00 – 16:00 (isRestrictedTime = True) any access requested by customers are denied.*

To check the authorization modality conflict, we specify a temporal logic property. The property checks that when a customer requests access to a resource, it leads to granting access to that, i.e., (isCustomer = TRUE) ~> (canCustomerAccess = TRUE). The verification of the property results in its violation and a counterexample is provided. More specifically, the model checker identifies a system behaviour where someone is a customer and requests access to a resource during a restricted time. The result is for access to be denied since canCustomerAccess = False, isCustomer = True and isRestrictedTime = True.

6 Conclusion and Future Work

In distributed systems and networks, especially in those of large-scale, the existence of policy conflicts is unavoidable. On the other hand, policy conflicts must be resolved as soon as possible since they can create problematic situations in the operational environment. In this paper, towards the detection of policy conflicts at the widest possible extend, we propose a new policy conflict categorization that covers most types of policy conflicts. Our categorization provides a global integrated view of

policy conflicts from various perspectives. Understanding in depth the policy conflict types, not only facilitates the conflict detection process, but also the conflict resolution that may have special characteristics per conflict category. Moreover, we demonstrated how model checking can be used to specify example policies and identify relevant conflicts.

Towards the direction of the evaluation of the proposed model in a real distributed system and network environment, which is part of our future work, we consider constructing several policies sets and the relationships between them. Separate policies must be developed for each examined perspective. Top-level policies must be defined and then a repetitive policy refinement process must be enforced down through the management structure. The refinement process starts from the top level, which concerns business goal requirements, which are refined to policies that mainly concern the Service Level Agreement (SLA) rules. This process continues up to the lower-level policy rules, which concerns specific software components or network elements.

We would like to emphasize on the fact that the above-described process of developing sets of policies in combination with the proposed integrated framework of policy conflict categorization, will provide administrators and engineers with adequate tools for minimizing the operational problems in distributed systems and networks. According to our knowledge this is not obvious for all other approaches presented in the related work section.

In addition, our future work includes the investigation of potential conflict resolution mechanisms per policy conflict category, as well as the evaluation of their applicability in real operating environments. In addition, we intend to implement policy conflict detection mechanisms for multi-controller software-defined network which is suggested by Lu et al. (2019). Finally, its effectiveness has been tested in their campus network experimentally.

References

Aiken B. et al, (1999), "Terminology for Describing Middleware for Network Policy and Services", Internet draft-aiken-middleware-reqndef-00.txt, April 30, 1999

Al-Jawad A., Shah P., Gemikonakli O. and Trestian R. (2018), "Policy-based QoS Management Framework for Software-Defined Networks", Conference: 2018 International Symposium on Networks, Computers and Communications (ISNCC), Available at: https://bit.ly/3b9qMhY (accessed 5/5/2020)

Al-Shaer E., Hamed H., (2003) “Firewall policy advisor for anomaly discovery and rule editing”, Proceedings of IFIP/IEEE International Symposium on Integrated Network Management, March 2003.

Alquhayz H, Alalwan N., Alzahrani A. I., Al-Bayatti A. H. and Sharif M. S. (2019), “Policy-Based Security Management System for 5G Heterogeneous Networks”, Wireless Communications and Mobile Computing/2019/ Research Article | Open Access Volume 2019 |Article ID 4582391, Available at: https://bit.ly/3ddLwGS (accessed 6/5/2020)

Bandara A., Lupu E., and Russo A., (2003) “Using Event Calculus to Formalise Policy Specification and Analysis”, Proceedings of 4th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2003), pp. 26-39, IEEE Computer Society, Lake Como, Italy, June 2003.

Boiten E., Bowman H., Derrick J., Linington P., Steen M., (2000), “Viewpoint consistency in ODP”, Computer Networks, Vol. 34, Issue 3, pp. 503-537, September 2000

Chadha R., Kant L. (2008) “Policy-driven in mobile ad-hoc network management”, Willey series in Telecommunication and Signal Processing, 2008

Charalambides M. et al., (2005), “Policy Conflict Analysis for Quality of Service Management” in Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks POLICY'05, Stockholm, Sweden (2005) 99-108

Charalambides M., Flegkas P., Pavlou G., Rubio-Loyola J., Bandara A.K., Lupu E., Russo A., Dulay N., Sloman M. (2009), “Policy Conflict Analysis for DiffServ Quality of Service Management,” IEEE Transactions on Network and Service Management, 2009, vol. 6, pp. 15-30

Chen D., Vernadat F., (2007), “Standards on enterprise integration and engineering—state of the art” pp. 235-253 | Published online: 19 Feb 2007, https://bit.ly/2YxA9p4 (accessed 28/3/2017).

CSIMQ, Article 62, Issue 11, June/July 2017, Pages 1–19, Available at: https://bit.ly/3feNQPz (accessed 4/5/2020)

Dimitrios Kallergis, Zacharenia Garofalaki, Katsikogiannis G., Douligeris C., (2020), “CAPODAZ: A Containerised Authorisation and Policy-driven Architecture using Microservices”, to appear in Ad Hoc Networks, Elsevier

Dumitrescu C., Foster I. (2004), “Usage Policy-based CPU Sharing in Virtual Organizations”, Proceedings of the 5th International Workshop in Grid Computing, 2004

Dunlop N., Indulska J., Raymond K. (2003), “Methods for conflict resolution in policy-based management systems”, Proceedings of the 7th

International Conference on Enterprise Distributed Object Computing (EDOC 2003), Brisbane, Australia, 2003.

Farooqui, K., Logrippo, L., de Meer, J. (1995), "The ISO Reference Model for Open Distributed Processing – An introduction", Computer Networks and ISDN Systems, Vol. 27, 1995, pp. 1215-1229.

Foresti S. and Persiano G., (2016), "Cryptology and Network Security", 15th International Conference, CANS 2016 Milan, Italy, November 14-16, 2016 Proceedings, Available at: https://bit.ly/2WnbTTW (accessed 5/5/2020)

Giebas D., Wojszczyk R., (2020), "Deadlocks Detection in Multithreaded Applications Based on Source Code Analysis", Appl. Sci. 2020, 10, 532; doi:10.3390/app10020532, Available at: https://www.mdpi.com/journal/applsci (accessed 4/5/2020)

Grompanopoulos, C., Gouglidis, A., & Mavridou, A., (2021), "Specifying and verifying usage control models and policies in TLA+". International Journal on Software Tools for Technology Transfer, 1-16.

Haded H., Al-Shaer E. (2006) "Taxonomy of conflicts in network security policies" IEEE Communications Magazine, 44(3):134-141, March 2006.

IETF, (2005), The IETF Policy Framework Working Group: Policy Charter available at: https://bit.ly/2SxPHpc (accessed 29/03/2017)

Jajodia S., Samarati P. et al., (1997), "A Logical Language for Expressing Authorisations", IEEE Symposium on Security and Privacy, Oakland, 1997.

Katsikogiannis G., D.Kallergis, Z.Garofalaki, Mitropoulos S., Douligeris C., (2018), A policy-aware Service Oriented Architecture for secure machine-to-machine communications', Journal of Ad Hoc Networks, pp. 70-80, November 2018, Elsevier Science Publishers.

Katsikogiannis G., Mitropoulos S., Douligeris C., (2013), "Policy-based QoS management for SLA-driven adaptive routing," IEEE Journal of Communications and Networks (JCN), July 2013, vol.15, pp. 301-311

Kuhlisch R., (2017), "Modeling and Recognizing Policy Conflicts with Resource Access Requests on Protected Health Information", Complex Systems Informatics and Modeling Quarterly

Lamport,L. (accessed on March 2020), "The TLA+ Home Page". Available at: https://bit.ly/2SuvoZy

Linington P., Milosevic Z., Tanaka A., Vallecillo A., (2011), "Building Enterprise Systems with ODP: An Introduction to Open Distributed Processing", Chapman & Hall/CRC, 2011.

Lu Y., Fu Q., Xi X., Chen Z., Zou E. and Fu B. (2019), "A policy conflict detection mechanism for multi-controller software-defined networks."

International Journal of Distributed Sensor Networks. https://bit.ly/2YyIzMP (accessed 5/5/2020)

Lupu E., and Sloman M., (1999), "Conflicts in Policy-based Distributed Systems Management", IEEE Transactions on Software Engineering vol.25(6), pp. 852-869, 1999.

Moffett J., Sloman M. (1994) "Policy Conflict Analysis in Distributed Systems Management", Journal of Organizational Computing, Vol.4, No.1, 1994, pp.1-22

Moffett J., Sloman M., (1993), "Policy Hierarchies for Distributed Systems Management",IEEE JSAC Special Issue on Management, Vol.11, pp.1404-1414, 1993.

Paoli D. F., Schulte S. and Johnsen B. E., (2017), "Service-Oriented and Cloud Computing", 6th IFIP WG 2.14 European Conference, ESOCC 2017 Oslo, Norway, September 27-29, 2017 Proceedings, Available at: https://bit.ly/2YwX1ow (accessed 4/5/2020)

Phan T., Han J., Schneider J.-G., Ebringer T. and Rogers T., (2008) "A Survey of Policy-Based Management Approaches for Service Oriented Systems", in Proceeding of the 19th Australian Conference on Software Engineering (ASWEC '08), Perth, Australia, March 2008, pp. 392-401

Pisharody S. (2017), "Policy Conflict Management in Distributed SDN Environments", A Dissertation Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy, Available at: https://bit.ly/2YDJgEQ (accessed 6/5/2020)

Raymer D., Strassner J., Lehtihet E., (2006), "End-to-End Model Driven Policy Based Network Management", in Proceeding of the 7th IEEE Workshop on Policies for Distributed Systems and Networks (POLICY '06), Western Ontario, London, Canada, June 2006

Reiff-Marganiec, S., Turner, K.J. (2004), "Feature interaction in policies", Computer Networks, 45(5):569–584, 2004.

Stone G.N., Lundy B., and Xie G.G., (2001), Network "Policy Languages: A Survey and a New Approach", IEEE Network, January/February 2001.

Syukur, E., Loke, S.W., and Stanski, P., (2005), "Methods for Policy Conflict Detection and Resolution in Pervasive Computing Environments", Proceedings of Policy Management for Web Workshop in Conjunction with WWW2005, Chiba, Japan.

Valenza F., (2017), "Modelling and Analysis of Network Security Policies", Doctoral Dissertation Doctoral Program in Computer Engineering (29th cycle), Available at: https://bit.ly/3b18EXB (accessed 4/5/2020)

Warner T., (2018), “10 Best New Features in Windows Server 2016”, Business News Daily Small Business Solutions and Insoiration, Available at: https://bit.ly/2YxRYo8 (accessed 4/5/2020)

Figure 1: A classification of policy conflicts

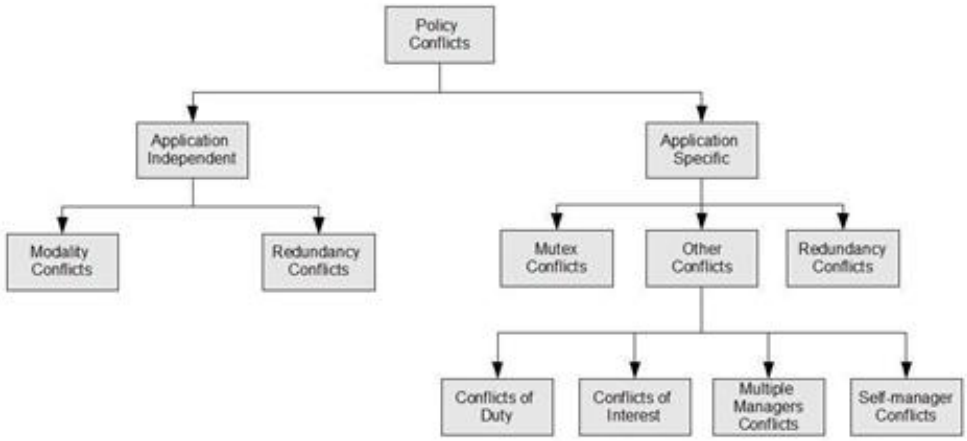


Figure 2: The integrated approach in policy conflicts categorization

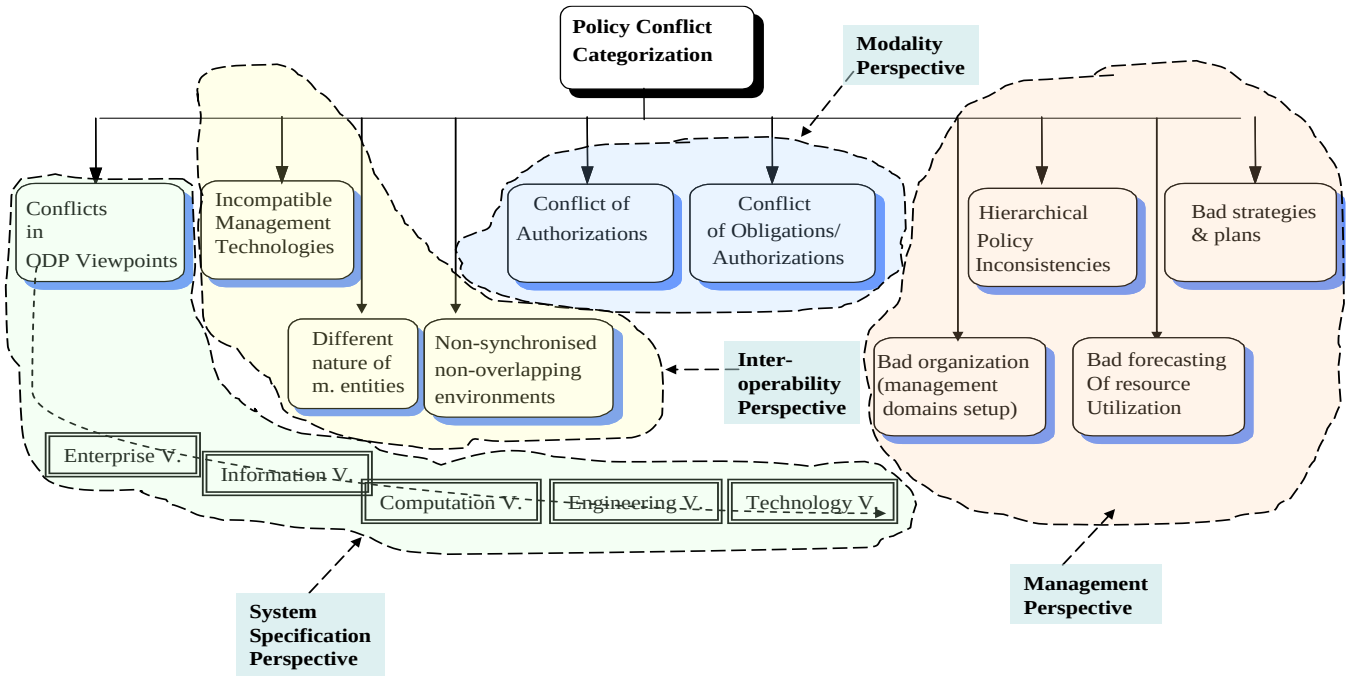
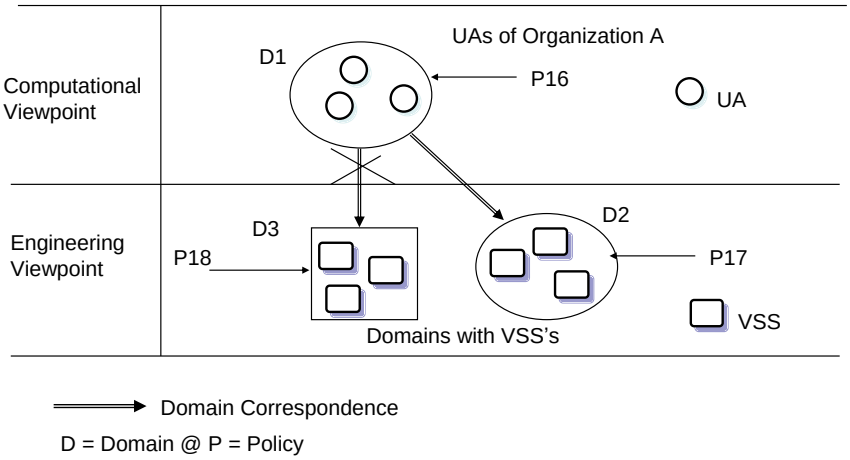


Figure 3: Domain Correspondence in ODP viewpoints



Policy Types	Domain Entity	Roles	Policy Relation	Modalities
Unconditional Goal	Multiple Entities Different Domain	Multiple Entities Multiple Relations	Refinement	Authorization
Conditional Goal	Multiple Entities Different Branch	Multiple Entities Single Role	Relaxation	OPI
Unconditional ECA	Multiple Entities Same Branch	Single Entity Multiple Roles	Independence	Preferences
Conditional ECA	Single Entity	Single Entity Single Role		Temporal

Table 1: Dimensions of policy conflicts as identified by Reiff-Marganiec and Turner in (Reiff-Marganiec and Turner, 2004)

Appendix A: Modality conflict in PlusCal

```
------------------- MODULE ModalityConflict ---------------

EXTENDS TLC, Integers
(* --algorithm flags

variables
    Hour \in BOOLEAN,
    Bw_lt_1mb \in BOOLEAN,
    Backup_done \in BOOLEAN,
    Role_is_accountant \in BOOLEAN;

begin

    Hour := RandomElement({TRUE,FALSE});
    Bw_lt_1mb := RandomElement({TRUE,FALSE});
    Role_is_accountant := RandomElement({TRUE,FALSE});

    Backup_done := FALSE;

    if (Hour = TRUE /\ Role_is_accountant = TRUE) then
        if (Bw_lt_1mb = FALSE) then
            Backup_done := TRUE;
        elsif (Bw_lt_1mb = TRUE) then
            Backup_done := FALSE;
        end if;
    end if;

end algorithm; *)

Property_Modality_01 ==  ( (Hour = TRUE /\ Role_is_accountant
= TRUE)  ~> (Backup_done = TRUE) )

=============================================================
```

Appendix B: Authorization modality conflict in PlusCal+

```
---------------- MODULE AuthorizationModality -------------

EXTENDS TLC, Integers
(* --algorithm flags

variables
    isCustomer \in BOOLEAN,
    canCustomerAccess \in BOOLEAN,
    isRestrictedTime \in BOOLEAN; \* Time is 14:00 - 16:00

begin
    canCustomerAccess := TRUE;
    isCustomer := TRUE;

    while TRUE do

        isRestrictedTime := RandomElement({TRUE,FALSE});
        isCustomer := RandomElement({TRUE,FALSE});

        if (isRestrictedTime = TRUE) then
            canCustomerAccess := FALSE;
        elsif (isRestrictedTime = FALSE) then
            canCustomerAccess := TRUE;
        end if;

        if (isCustomer = TRUE) then
            canCustomerAccess := TRUE;
        elsif (isCustomer = FALSE) then
            canCustomerAccess := FALSE;
        end if;

    end while;


end algorithm; *)

Property_Authorization_01  ==  (  (isCustomer  =  TRUE  )    ~>
(canCustomerAccess = TRUE) )

=============================================================
```