On Operational Policy Conflict Detection and Resolution in CPS-IoT systems

Jared Hall
dept. Computer and Information Science
University of Oregon
Eugene, Oregon
jhall10@cs.uoregon.edu

***Abstract*—With the convergence of the field of Cyber-Physical Systems (CPS) and the "Internet of Things" (IoT), a new generation of systems, dubbed CPS/IoT systems [1], have come about. These systems focus on controlling physical environments semi or even fully autonomously based on an operational policy designed by the user for their environment. However, conflict in the decision-making processes these systems employ can easily happen. These conflicts arise due to a confluence of factors such as the distributed nature of CPS-IoT systems, changes in user preferences, or the use of multiple intelligent sub-components. For this work, we will be investigating the current research concerning the problem of Operational Policy conflict detection and resolution for CPS-IoT Systems. This survey will include the current state-of-the-art in conflict detection in CPS-IoT systems as well as a brief overview of conflict detection methods in related fields such as distributed systems and Software-defined networks. Our aim with this analysis is to present a summary of the key insights, objectives, and methods of addressing policy conflict detection and resolution pioneered by these fields.**

I. Introduction

With the convergence of the field of Cyber-Physical Systems (CPS) and the "Internet of Things" (IoT), a new generation of systems, dubbed CPS-IoT systems [1], have come about. These systems focus on giving the user the ability to build ambient intelligence into physical environments via the use of a collection of rules/procedures that we call "Operational Policy" [2], [4]. In the most general sense, an "Operational Policy" is an abstraction of a plan for the moment-to-moment operation of a CPS-IoT system. The policy maps out what actions the system should take in a given instance and what constraints it should obey [2]. Typically, operational policies take the form of a collection of rules or procedures that are designed by the user when they provision their CPS-IoT system and are then enforced by said system.

However, conflict in the decision-making processes these systems employ can easily happen. These conflicts arise due to a confluence of factors such as the distributed nature of CPS-IoT systems, changes in user preferences, or the use of multiple intelligent sub-components (for instance intelligent agents, inference components, etc.). Moreover, many IoT entities send telemetry or execute actions as atomic, infrequent, transactions due to power/computing constraints and the active duty cycle. This leaves them vulnerable to instances where multiple controllers send commands in order, but they are received by the actuator simultaneously. There may also be conflicts that spawn from the collaboration between edge systems and cloud-based systems [3]. Lastly, in CPS-IoT platforms that use a deliberative AI algorithm to make decisions, situations, where multiple controllers send conflicting actions, may arise. Whether it is due to a mistake by the user, an intelligent sub-component, or due to the distributed nature of CPS-IoT systems, conflicts in the enforcement of an operational policy for a CPS/IoT control system can have severe consequences. For instance, one could deadlock critical components in a smart city by constantly spawning conflicts in its subsystems or breaking down equipment in a hospital by spamming actions. These "pain points" necessitate research concerning how to detect these conflicts before they cause damage and resolve them.

For this work, we will be investigating the current research concerning the problem of Operational Policy conflict detection and resolution for CPS-IoT Systems. This research will include the current state-of-the-art in conflict detection in CPS-IoT systems as well as a brief overview of research offerings in conflict resolution. However, it is our position, as will be shown in this paper, that the current state of the art does not sufficiently consider all conflict types that may arise as Artificial Intelligence and Machine Learning algorithms are more closely integrated into CPS-IoT control systems and as the field progressively moves toward cloud-edge collaborative systems.

The rest of this paper is organized as follows: in Section II, we present a brief overview of the background information needed to fully understand the research space. In section III, we begin our discussion of operational policy conflict by looking at what is currently done in related fields. This discussion is continued in Section IV by extending our review of a sample of related works in CPS-IoT. In section V, we discuss operational policy conflict resolution. In section VI, we discuss the open problems and challenges regarding operational policy conflict detection and resolution. Finally, we end in section VII with our final thoughts on the problem space.

II. Background

In order to gain a better understanding of the landscape of this research area, there is a need to clarify the definitions and relationships among a number of related concepts that are common in the literature.

A. CPS-IoT Systems

Formally, a 'pure' CPS consists of three core components: A cyber component that performs computation (e.g., information processing and control), a communication component that handles the communication between the cyber and physical components, and a physical environment that consists of physical entities or processes [2] as shown in Fig. 1 below.

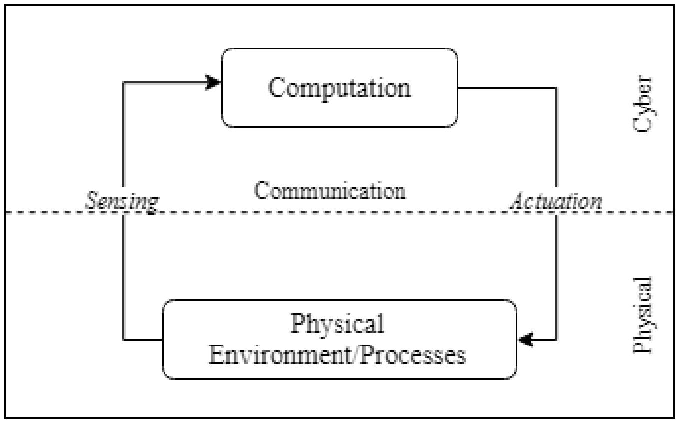

Fig. 1. Cyber-Physical Systems Component Model

In these systems, the cyber component is tightly coupled with the physical component via a central feedback loop known as the Transitional State Change (TSC) Feedback loop [1]. In this feedback loop, the physical environment informs the cyber component of its current state via the communication component, and the cyber component exerts control over the physical environment which alters its state, causing the cycle to repeat.

The two key advancements brought by CPS-IoT systems concern the integration of the IoT with the communication component and the inclusion of humans as a distinctly separate component [1]. The introduction of the IoT as a central component in CPS design has greatly expanded the scale and scope of these systems as well as broadly expanding the number of application domains for CPS-IoT systems [1]. Secondly, Humans play a central role both as entities in the physical environment and as users of the system (e.g., "human-in-the-loop") via their interactions with both the system and the physical environment. Because of this, humans are added as the fourth component in CPS-IoT systems, as stated by the National Institute for Standards and Technology (NIST) in [1]:

"… reflect the varying roles humans may have in CPS/IoT systems, ranging from user to component, environmental factor, etc. (for example, for a Level 3 automated vehicle a passenger is a user, a safety driver is a component, and a pedestrian is an environmental factor). The interactions of humans with CPS/IoT systems may be limited to the logical realm, to the physical realm, or extend to (and link) both. Because of this diversity of interactional modes, humans are treated as a distinct component in the CPS/IoT Components Model."

CPS-IoT systems can vary wildly in both architecture and purpose with applications to domains such as the Internet of Multimedia Things (IoMT) and the internet of healthcare things (IoHT). This extends to the types of entities in these systems as well, as noted in [1] - a dense heterogeneity of entities is one of the core differences between CPS-IoT systems and standard distributed computing systems.

B. Rules and Operational Policy

Among the technologies introduced to make using CPS-IoT systems easier to use, one of the most influential would be the introduction of the event-driven rules model to CPS-IoT systems [4]. These rules add a sort of reactive intelligence into the operating logic of the CPS-IoT system and simplify the cost of building the physical environment (i.e., the time and effort) by allowing the user to simply write a set of rules to govern the CPS-IoT system and then leave the enforcement of said rules to the system [5].

These rules come in the form of Event-Condition-Action (ECA) rules. According to [5], an ECA rule "autonomously reacts to actively or passively detected simple or complex events by evaluating a condition or a set of conditions and by executing a reaction whenever the event happens, and the condition is true". Formally, ECA rules consist of independent statements in the following format:

$$\textbf{ON}\ Event\ \textbf{IF}\ Condition\ \textbf{DO}\ Action \tag{1}$$

In CPS-IoT systems, an event is generated by "entities" or "things" in the physical environment (e.g., the temperature is 75 degrees). A condition is a Boolean trigger constraint on the rule itself (i.e., the rule will only fire if the condition is true). Whereas, an action is a control to be enforced on the entity to which the rule applies (e.g., turn on the fan). Some notable characteristics of ECA rules are:

1) A rule is activated only by events.
2) Its execution is autonomous and independent of other rules in the system.
3) It implements a reaction to the incoming event.
4) It contains a guarding condition to execute such actions

A crucial benefit of using rules instead of programming scripts to control the core operating logic of a CPS-IoT system is that they can be easily grasped by individuals with no background in computer science as ECA rules are already in common use by many application domains [5]. This ubiquity expands the user base of CPS-IoT systems while also giving the engineers of these platforms the ability to decouple the core operating logic from the enforcement system since rules can be altered on the fly without giving the user access to the underlying code.

The CPS-IoT systems presented in [3] are representative examples of many CPS/IoT systems that utilize ECA rules. This kind of approach is widely used in the IoT industry with platforms such as Amazon's IoT platform (https://aws.amazon.com/iot/) and Xively (https://www.xively.com/) and in CPS-IoT systems research as seen in our own work [6]–[8]. These systems come with a number of benefits, such as device virtualization, an elastic cloud architecture that allows users to change/create building blocks in their service definition on-demand [2]. They also contain the ability to dynamically alter the rule set for each system by allowing the user to manually write the rules and inject them into the CPS-IoT system.

These kinds of systems were the first to feature a decoupling of the operational logic from the code itself, allowing users to change the system on the fly increasing the elasticity of the CPS-IoT system itself. These types of systems often come with dynamic rule engines or other artificial intelligence agents [3], that decouple the operational logic from the system itself giving it robust elasticity while also making the platform itself reusable for different application domains.

C. Conflict Detection vs. Management vs. Resolution

In order to clearly define the area around which this research revolves, we will take a brief look at three closely related, albeit distinct, areas of operational policy conflict: Conflict detection, conflict resolution, and conflict management. According to [9], conflict detection is a process that involves determining whether or not a conflict has or may occur. This can be done by analyzing the flow of events at the end device or statically analyzing the rule set for potential conflicts. Following an occurrence of a conflict, conflict resolution involves resolving the conflict in a way that mitigates damage to the system. Principally, both dynamic conflict detection and resolution are reactive processes that begin after a conflict has occurred.

By contrast, conflict management is a process that begins before a conflict has occurred. The general idea for conflict management involves designing and managing the system in such a way as to mitigate potential conflicts before they occur (e.g., by employing avoidance or proactive relaxation of the rule sets as done in [4]). This may take the form of a policy language with a cohesive set of design principles that mitigate potential conflicts or a framework/architecture that enforces strict relational boundaries between dependent entities.

However, regardless of how proficient the operational policy conflict management process is for a CPS-IoT system, conflicts will still occur. The best we can aim for is to mitigate the largest number of potential conflicts with a well-designed conflict management system, quickly detect conflicts that do occur with a well-designed conflict detection system, then resolve them in a way that presents the least harm to the system. As will be discussed in the next section, great strides have been made in adjacent fields addressing operational policy conflicts such as distributed systems and software-defined networks with some work being done for CPS-IoT systems.

III. On conflict detection in the Literature

In this section, we will present an analysis of the work done in related fields regarding policy conflict detection. Our aim with this analysis is to present a summary of the key insights, objectives, and methods of addressing policy conflict detection pioneered by these fields. This analysis will allow us to gain an understanding of how operational policy conflicts are currently being detected and will allow us to approach the topic of conflict resolution with a solid base of understanding. We will start this analysis with an overview of research from the field of distributed systems.

A. Analysis of Conflict detection in Distributed Systems

In the field of distributed systems, the problem of detecting and resolving conflicts in instructions sent between systems is an old problem with some of the first publications regarding this problem being made in the 1990s by M. Sloman et. al. [9]–[11]. At its core, CPS-IoT systems are distributed systems. As such, reviewing the research done by M. Sloman and N. Dunlop [12], [13] may give us valuable insight into how to approach this problem for CPS-IoT systems.

In [16], Sloman et. al provides a number of useful definitions for what they consider policy in the distributed systems space. ”Policies are rules governing the choices in the behavior of a system” [9]. According to Sloman et. al [16], [18], the following are the main types of policies in a distributed system: **Obligation policies** are ECA rules that can be used to define adaptable management actions. These policies thus define the conditions for performing a wide range of management actions such as changing the Quality of Service, when to perform storage server backups, registering new users in a system, or installing new software. **Authorization policies** are used to define what services or resources a subject (management agent, user or role) can access.

The general idea presented in [10] is to take action as far as possible on the basis of general policies, not of particular cases. This implies the generation of policies that apply to abstractly defined situations, and to groups of components and users of the system rather than individual units. For example, the same policy may apply to all people in a department or to the set of files pertaining to an application. This idea is further expanded in [19] with the introduction of policy roles. The general idea of policy roles is to constrain potential conflicts of modalities by assigning an object a policy role or a relationship role. This role could be inherited by sub-classes, allowing for similar objects to share policies.

In [17], Sloman et. al further expands upon their previous work by introducing the idea of hierarchical policies. This innovation allows the user to specify something similar to a meta-policy that can be broken down into specific sub-policies automatically by the policy management system. This work also introduces the idea of management action policies which we believe is a precursor to the concept of ”operational” policy found in CPS-IoT today. The authors further define the types of policies that are of core interest to conflict analysis in distributed systems: ”**Management action policies**

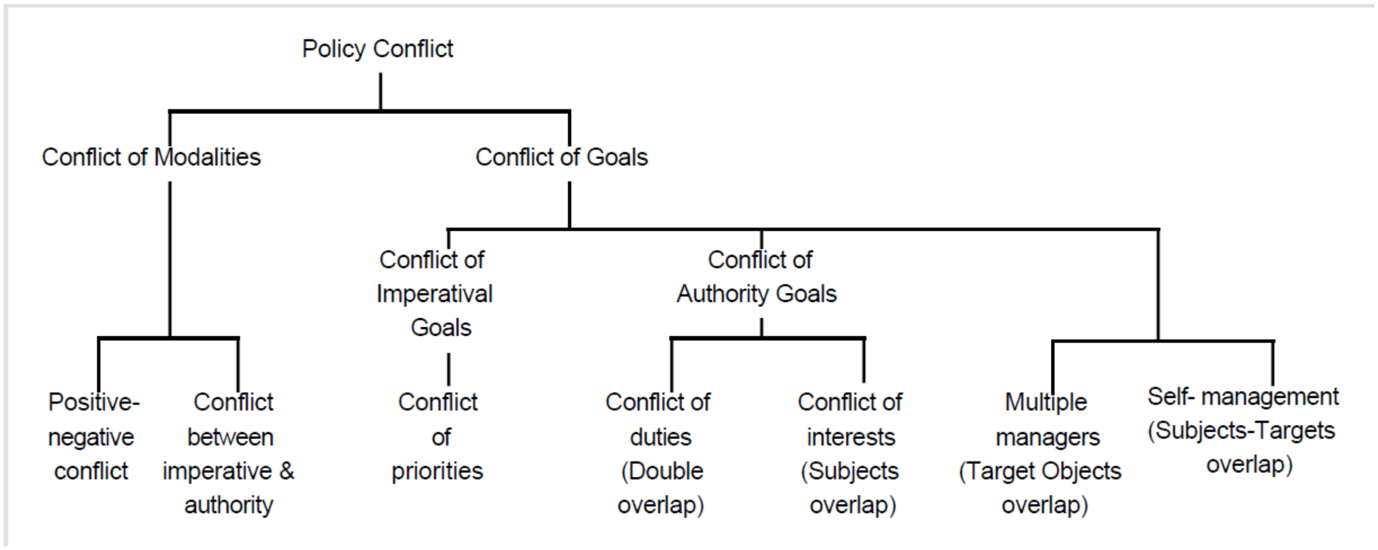
Policy Conflict
Conflict of Modalities
Conflict of Goals
Conflict of Imperatival Goals
Conflict of Authority Goals
Positive-negative conflict
Conflict between imperative & authority
Conflict of priorities
Conflict of duties (Double overlap)
Conflict of interests (Subjects overlap)
Multiple managers (Target Objects overlap)
Self- management (Subjects-Targets overlap)

Fig. 2. A graph representation of the different types of policy conflicts from [11]

are the main kind which are of interest to distributed system management; briefly, they describe a persistent, positive or negative, imperative or authority for a set of policy subjects to achieve goals or actions on a set of target objects...” [11]. It is plain to see how our modern view of operational policy for CPS-IoT systems was derived from the concept of Management Action Policies of 30 years prior.

In general, there seems to be no singular definition of ”policy conflict” from our review of the literature, instead, the common approach is to present a list of conflict classes. The common classes of policy conflict we have seen can largely be broken down into two umbrella classes: **Static** and **dynamic**. In [12], [13], the authors define a static conflict as ”a policy conflict that may occur from conflicting rule sets during compile time”. Static policy conflicts can be resolved at the time when a rule set is compiled by applying a wide variety of methods such as relaxing the rule-set as explained in [12]. Whereas a dynamic, potential conflict is quite unpredictable, in that it may, or may not, proceed to actual conflict; that is, the inconsistency may be exposed temporarily, or indeed not at all [12]. To this day, dynamic policy conflicts represent the more difficult of the two types to actually detect and resolve.

The concept of Policy conflict was expanded in [11]–[13] to include additional classes of policy conflicts in distributed systems as seen in Fig. 2. The major distinction is between conflict of modalities and conflict of goals. Put simply, a conflict of modalities refers to whether or not an entity is forbidden to execute an action, whereas a conflict of goals occurs when an entity is told to execute actions with conflicting ends. Conflicts of modalities can be recognized without reference to the meaning of the policy goal, whereas conflicts of goals depend upon the semantics of the goal, or are application-dependent. This is further expanded upon in [11] with the following general types of modality conflicts:

1) **O+/O-** the subjects are simultaneously required and required not to perform the same actions on the target objects.
2) **A+/A-** the subjects are both authorized and forbidden from performing an action on the target objects.
3) **O+/A-** the subjects are required but forbidden to perform the actions on the target objects (obligation does not imply authorization in our case).

To give a brief example, consider a simple distributed system where we have a server, a client, and an HVAC controller. The server has a policy that the controller is to be turned off after working hours, whereas the client can send commands to the controller at any time. In this setup, a conflict of modalities would occur if the client has the policy to turn the controller on outside of working hours since from the controller’s perspective it would be both forbidden and obliged to execute an action. A conflict of goals would occur if the server tells the HVAC controller to turn on to heat the room to 75 degrees while the client says to heat the room to 80 degrees. In this instance, both the client and the server agree that the HVAC controller should be turned on but they disagree on the end goal of what temperature to reach. In particular, this would be known as a conflict of duties since we would have a double overlap between the subject (both policies are about the HVAC system) and the actions to be taken (turn it on for a period of time).

In [9], [20], Lupu et al. describe conflict analysis for management policies, using a tool to conduct offline detection of conflicts in a large-scale distributed system. Sloman et. al further expanded their works with Ponder [14] and Ponder2 [15] which provide a common means of specifying policies that map onto various access control implementation mechanisms for firewalls, operating systems, and databases. It advocates for a de-centralized model of autonomous agents cooperating with each other and composing into more complex configurations. The ponder systems were among some of the first works in developing ontology-based approaches which use a formal language to specify the policy which allows for programmatic answers to conflict detection and resolution.

In these papers, Sloman et. al presents many of the categories and primitives used to discuss policy-based distributed systems today. In this work, we will be using many of the same ideas originally presented in these works while also adapting them to the field of CPS-IoT.

B. On Policy Conflict in Software-Defined Networks

Next, we will give a summary of some of the works regarding policy conflict in the field of Software-Defined Networks (SDN). There have been several attempts to classify policy conflicts in the field. In [21], the authors deal with the packet classification and filter conflict detection problem. They use a KD tree [22] to verify if two rules apply different actions on the same packet. This misses out on some conflict classification types, that involve sub-optimal rules. Fu et al. [24] manage policies as they apply to IPSec tunnels in both inter- and intra-domain environments. Hari et al. [23] present a conflict detection and resolution algorithm using a k-tuple filter that grows linearly.

FAME [25] is a conflict management environment to detect and resolve conflicts by using rule-based segmentation. In FAME, the authors used a matrix to represent conflicting and non-conflicting address segments; but fails while trying to represent larger rule sets. In [26], the authors proposed a formalization of conflict detection for firewalls, but constrained themselves to only look at rules where the actions are different; as such, Fame cannot address conflicts of goals as discussed earlier.

Rei [27] is a language based on deontic logic [28] that defines security policies as possible actions on a resource. All policies in Rei are free of conflicts due to the presence of meta-policies defined by an administrator, which are used to resolve conflicts. If a meta-policy that covers the conflict does not exist, by default the deny action is prioritized. While this method may have sufficient results in SDN, we have our doubts about its usefulness in CPS-IoT as the dense heterogeneity of the domain would lead to meta-policies just as complex as the base operational policy. We do however think that an investigation into a simpler representation of the operational policy for CPS-IoT is worthwhile. We have found some works in CPS-IoT that adopt a similar approach while not quite leaning into a full meta-policy approach.

Following this idea, Fang [29] is a tool that reads vendor-specific configuration files and converts them into an internal representation, which is then presented to the administrator in a tabular form in simple text. While it is a step in the right direction, it does not display any relation between conflicting rules. The onus is on an experienced administrator to submit the right query that would present the conflict. PolicyVis [30] used overlapping bars to represent conflict types, and colors to represent the action. However, the conflicts are visible only when a certain scope is defined. A sunburst visualization is used in [31] to visualize the rule set but does not provide any visualization for flow rule conflicts. None of the above works, provide scalable rule conflict visualization that provides high-level conflict categorization, with granular information provided to the administrator.

Pyretic [32], is a high-level language written in Python courtesy of the Frenetic project [33], which allows users to write modular applications. Modularization ensures that rules installed to perform one task do not override other rules. Using a mathematical modeling approach to packet processing, Pyretic compares the list of rules as functions that use a packet as an input, and have a set of zero or more packets as output. Given its mathematical base, Pyretic deals effectively with direct policy conflicts, by placing them in a prioritized rule set much like the OpenFlow flow table. However, indirect security violations or inconsistencies in a distributed SDN environment cannot be handled by Pyretic without a flow tracking mechanism such as the one discussed in [34].

VeriFlow [35] is a proposed layer between the controller and switches that conducts real-time verification of rules being inserted. It uses search rules based on Equivalence Classes (ECs) to maintain relationships and determine which policies would be affected in case of a change. Thus, it can verify that flow rules being implemented have no errors due to their dependence on faulty switch firmware, control plane communication, reachability issues, configuration updates on the network, routing loops, etc. Like VeriFlow, NetPlumber [36] sits between the controller and switches. Using header space analysis [37], it ensures that any update to a policy is compared to all dependent policies to prevent and report violations.

In [38], [39], the authors present a Hierarchical Flow Tables (HFT) framework that organizes SDN rules in an OpenFlow switches' flow tables (which is addressed as "policy" in this work) into trees. It then abstracts the OpenFlow switch's actions to ease the traffic engineering tasks like "guarantee minimum bandwidth", each tree node can independently choose the action to perform on a packet. Conflicts in different parts of the tree are resolved with user-defined conflict-resolution operators situated at each node of the tree. The target application is derived from the policy trees and the actions on traffic supported by HFT, which include: admission control, guaranteed minimum bandwidth, and "don't care".

Our brief overview of some of the recent work in SDN research reveals that the area tends to aim directly at conflict resolution at the price of constraining the application domain of the solution proposed. Each of these works chooses some specific applications that cause conflicts and tries to solve the problem in its own way. While this philosophy of approach has yielded useful results, we believe that this approach will prove difficult in CPS-IoT due to the heterogeneity of its application domains, potential policies, and CPS-IoT system configurations. For instance, a social-IoT system for home and building automation will look very different from an industrial-IoT system. In order to truly address a policy conflict in CPS-IoT we need to know: why it happens, what class it belongs to, which factors influence its occurrence, and how.

We assume a broader understanding of conflict to be essential for an effective holistic conflict-handling strategy. Therefore, we prefer a layered approach focusing on a synthesis of both static and dynamic conflict detection on a broad scale and taking the step toward further handling (resolution, avoidance, application design) based on a more thorough understanding of interference between applications' intents.

IV. Overview of Conflict Detection Methods in CPS-IoT

In general, CPS-IoT conflict detection methods can be described using the following four main classes: a) "Rule-Based", b) "Formal Methods-based" c) "Ontology-Based", and d) "Application-Based". In a rule-based system, CPS-IoT services are expressed as a coherent operational policy using a collection of rules, typically ECA rules, as discussed previously. Whereas methods that use a formal approach, rely on an inner calculus or logic-based procedure then look for contradictions in the operational policy to determine if a conflict has occurred. In an ontology-based system, the operational policy can be defined as a specification of the conceptualization of a domain. Ontology-based frameworks enable the representation of knowledge through the use of well-defined semantic and syntactic rules. Ontologies are collections of domain knowledge encoded using axioms, natural language labels, synonyms, and definitions. They facilitate the creation of reusable entities and relations among devices, events, and activities in a CPS-IoT system. In an ontology-based approach, a conflict is defined as an interference with the user's preferences or when post-conditions for initiating an event conflict with preconditions of a happening event. Application-based work is concerned with resolving inter-application conflicts; the presence of multiple apps controlling the same actuator or device results in potentially undesirable interactions. For instance, in a smart home, the owner has two apps: one that sounds an alarm when smoke is detected, and another that unlocks the doors. Additionally, the same owner installed an app that automatically locks the door when the homeowner leaves the house. Even after these apps were safely installed, they interact unexpectedly, resulting in inter-app conflicts.

A. *Analysis of Rule-Based Conflict detection Methods in the Literature*

In [49], the authors propose a static-conflict detection method that uses a generic knowledge graph model to represent the relationships between ECA rules defining IoT services and environmental properties. The proposed algorithms identify overlapping event pairs in the graph and the resulting direct and indirect conflicts. The algorithm performs well on a small number of real-world and synthetic data-sets, but the authors only evaluated their proposed work statically on predefined rules. The paper does not demonstrate the impact of rules in a real-time setting.

In [50], the authors present a dynamic conflict detection method that employs a framework for detecting and resolving conflicts via a weighted-priority scheduling algorithm. The proposed work captures and prioritizes events generated by heterogeneous systems in a smart home environment and utilizes a weighted scheduling algorithm to detect and resolve conflicts. If two ECA rules are in direct or indirect conflict, the event associated with the highest-priority rule condition is scheduled and triggered first. The work assigns a predefined priority to events and executes them accordingly. Although simple, choosing one of the conflicting events based on a fixed priority is not satisfactory since the appropriate action often does depend on the overall context. As such this method would struggle with a "conflict of goals" as discussed in section 3.1.

In [51], the authors developed a framework for detecting ECA mashup service conflicts in a smart home environment. The proposed mechanism defines a context descriptor for each mashup service instance that includes the following properties: Instance IoT service, which performs actions when the mashup service is processed; Context condition, which is impacted by the instance; and Direction, i.e., the direction of context change. As an illustration, consider the use of an air purifier as a mashup service for the home environment. When it is activated, indoor air quality improves. "Air purifier" is an instance, "indoor air condition" is the context, and "improves" is the direction in this example. The service conflicts are then identified by tracking context descriptors associated with the mashup service chains. The primary disadvantage of the proposed work is that the service conflicts are detected statically via context descriptors; the proposed work does not address dynamic conflict detection.

The authors of [52], [65] proposed a static conflict detection module that identifies predefined relationships between rules and uses these relationships to classify rule conflicts. A rule database is defined in an XML format and includes a rule table, a location tree, and an authority tree to filter out useless information to further improve detection efficiency. The work proposed achieves higher efficiency in comparison to that of other similar works. However, the work is based on static rule checking of simple rules and is incapable of detecting conflicts between complex policies. Moreover, the rule-parsing technique frequently produces inefficient results in recognizing actual conflicts.

Likewise [52], [65], [66], [66], [67], propose a rule verification mechanism based on a probabilistic analysis to determine the possibility of rule conflicts and anomalies. There are many studies on detecting conflicts in a Home Network System (HNS), as presented in [54]–[56], and [57], where each appliance was modeled as an object consisting of properties and methods. The behavior of an appliance is defined by state transition rules that include pre and post-conditions. Feature interactions are defined as conflicts between integrated services. These works consider feature interaction conditions in Linear Temporal Logic (LTL) formulas and validate them against HNS services. They consider both offline and online feature interactions. [56] proposed a classification of these feature interactions and resolution schemes for each type of interaction.

B. Analysis of Formal Methods-based Conflict Detection Methods in the Literature

The works seen in [68], [69] were based on a core calculus for IoT Automation that generalizes ECA rules for home automation apps. IOTA [68] involved the development of the first calculus for the domain of home and building automation. The evaluation part of that work includes conflict detection as one of the possible uses of the calculus by developing an analysis for determining when an event can trigger two conflicting actions based on the IOTA core formalism and a study for determining the root cause of an event's (non)occurrence. Their static conflict detection framework is based on a model checker that takes all the rules and, upon detecting a conflict, generates traces that detail sample execution, including the environment's initial state, the event that triggered the conflict, and the rules involved.

Another formal method similar to IOTA was presented in [70] named as dT-calculus and in [71] called mCWQ . The dT-calculus is a formal method for describing a distributed mobile real-time IoT system. The authors go over other calculi in this paper, such as Time pi-calculus and d-Calculus. Time pi-calculus can specify timing properties; however, time pi-calculus does not have direct specifications of action execution time and process mobility, such as ready time, timeouts, and deadline. d-Calculus allows only a simple type of temporal requirement to be specified, for instance, a temporal constraint on the minimum and maximum values. mCWQ is a mobility calculus that demonstrates how to capture the feature of node mobility and improve communication quality.

In [72], [73], the authors propose a semi-formal model, IRIS (Identifying Requirements Interactions using Semi-formal methods) for defining and detecting policy interactions/conflicts in intelligent homes. The authors introduced an interaction taxonomy that examined interactions between policy features other than telecommunications features in various domains. A run-time module for detecting and resolving policy conflicts is defined based on simulation techniques. IRIS's conflict detection work considered negative impact conflicts, in which one attribute or feature hurts another, and override conflicts, in which one attribute or feature overrides and cancels out the other. The disadvantage of IRIS is that it does not account for rule dependency and thus cannot detect direct and indirect dependence conflicts.

The authors of [41] proposed a formal model approach for dynamic conflict detection that detects conflicts within the defined rule-set of the IoT system that violate the safety properties defined in the system. The proposed work detects conflicts immediately after an event occurs, which may trigger an action or a series of actions that result in conflicts, and also demonstrates how conflicts can result in additional actions being executed, which eventually result in increased energy consumption.

In [74], the proposed framework detects conflicts as changes in the environment state that result in an undesirable application or user context, as expressed by a Constraint Satisfaction Problem (CSP). Based on the proposed conflict taxonomy and a CSP, the authors proposed a framework where the system context is represented by a set of variables and constraints. The system for detecting and resolving conflicts consists of four major modules: a context querier, a CSP solver, a solution translator, and a decision module. The context querier module receives context information and any number of context-related queries, all of which are described as environmental conditions. It then combines the context queries as additional conjuncts and converts them to a constraint system for input to the CSP solver module. The solution translator will receive the results from the CSP solver and convert them to the environment model's output format. Finally, the decision module analyzes the data to determine: a) whether there is a conflict; b) whether the conflict is solvable or not; and c) whether the conflict is solvable through the environment or occupant modification, or a combination of the two.

C. Analysis of Application-based Conflict Detection Methods in the Literature

IA-GRAPH developed in [58] and IotSan in [79] present a studies inter-app conflicts in the smart home domain and adopts an approach in which an app's transitions are represented as Satisfiability Modulo Theory (SMT) formulas. Conflicts between multiple apps are then detected using an SMT solver. The advantage of using an SMT solver is that it enables the generation of model representations that accurately capture the interactions of device controls in an application's source code. However, the approach disregards complex application logic contained in conditional statements with multiple time and threshold values; instead, it considers only simple logic which involves changing the state of a device with a binary value, i.e., on or off.

There exist several works on detecting and resolving conflicts across the CPS-IoT services provided by multiple applications in the context of smart homes [43], [44], [58], [80], [85] and in the context of smart cities [42], [86]–[88]. Among these [80], [83], [86], [89] are good examples of applications that resolve conflicts by assigning priority to different services based on the domain or administrator's understanding of each service, among others. In [43], the authors propose a framework inspired by the conflict detector work first presented in [44], which is based on actuation graphs. Actuation graphs provide a polymorphic abstraction of IoT actuators and sensors, which is then used to formulate remedial actions for a given IoT policy conflict. This framework is evaluated on a set of SmartThings apps in which sensor and actuator states are binary, i.e., on or off, and the set does not take into account any thresholds associated with the sensor or actuator state.

The original framework shown in [44] uses a directed graph where each node represents a collection of device (or module) states, and each directed edge represents a ECA relationship. Further, a node may contain a cascade of conditions (actions) rather than just one module state. A conflict occurs when a set of compatible conditions (i.e., conditions that can be satisfied concurrently) result in mutually exclusive states of the same device. However, the framework is unable to deal with policies that represent complex interactions between smart devices and users.

D. Analysis of Ontology-based Conflict Detection Methods in the Literature

In [45], the authors proposed a mechanism for detecting conflicts based on ontology using incomplete rules with a three-part scheme for resolving the conflicts. The first step is rule decomposition, which considers conjunctive clauses and their corresponding disjunctive normal forms to minimize rule conflicts. Following that, rule relationships are established, and finally, rule conflict incompleteness is determined using the relationships. Additionally, this method incorporates rule integrity to ensure that the sensor range of undefined actions is not exceeded. Experiments demonstrate that this method improves the accuracy of conflict detection.

The authors in [46] designed a conflict ontology model that represents different types of conflicts. A hybrid conflict detection algorithm is proposed by combining both knowledge and data-driven approaches to detect conflict among IoT services in multi-resident smart homes. [59] propose a conflict detection and resolution framework that performs context analysis based on an ontology that formally represents the environment's conditions.

E. Additional Static and Dynamic Conflict Detection Methods

In [82], [94] the authors developed a static analysis tool for tracking sensitive data flows as well as finding protection and safety issues in an IoT application. The developed system converts the source code of an IoT application (SmartThings app) to an intermediate representation, extracts a state model from this intermediate representation, and performs model checking on IoT applications to identify property violations, which indicate whether or not an IoT application adheres to predefined safety and security properties.

Watchdog, the dynamic conflict detection method proposed in [95], presents an architecture for detecting and resolving rule conflicts in the context of smart cities based on simulation and a topology of conflicts that can be detected at runtime. Watchdog identifies various characteristics of smart city services that contribute to potential conflicts in smart cities, including uncertainty, real-time, dynamic behavior of services, and Spatio-temporal constraints. Along with conflict detection and resolution, Watchdog's extension work, CityGuard [42], specifies additional safety requirements for a smart city. Most of these works presuppose prior knowledge of the system's components and the rules governing its evolution. They do not propose conflict resolution strategies that are both local and reusable. On the contrary, they employ globally scalable identification and resolution mechanisms.

V. On Operational Policy Conflict Resolution Methods

There are two basic ways of handling operational policy conflicts as seen in the literature: a) Avoidance, where potential conflicts are identified via static analysis (i.e., at design or initial configuration time) and resolved by changing the operational policy of the system, and b) Resolution, where the conflicts are detected at run-time and actions are taken to ameliorate them [4]. Static analysis tools build a semantic model of the software at compile time without executing it and then check various properties of the model as discussed in Section 3. Static conflict resolution is performed offline, and it is necessary to identify all conflict types that must be detected, including conflicts that are evident from the policy specification and that are not evident from policy specification but arise as a result of policy dependencies. Because most subsystems are deployed incrementally, static checking cannot completely eliminate all conflicts because of the following factors: a) the subsystem's operation and interaction with others continue to evolve as devices are added, removed, or upgraded; b) static checking must consider all possible interactions, which can become computationally intractable as the number of device inter-dependencies grow; and c) many statically identified (potential) conflicts may require highly unlikely or even unrealizable scenarios, and thus trying to avoid their many perfectly reasonable sets of actions would thereby handicap the system substantially. Finally, in a multi-party environment, it is not even possible to do a static analysis unless all parties are willing to share their detailed operational semantics and agree to make the identified changes.

In contrast, Dynamic resolution can operate in multiple ways, the extreme case being where conflicts are checked dynamically for each action, and if a conflict situation is recognized, it is resolved by taking some action, often by simply blocking one of the conflicting actions or changing the associated rules. There are several advantages of such a method over static analysis: First, conflict detection is simplified in that we only need to check the relevant rules as well as the safety properties of the affected devices before taking action, and then block the action (or initiate a counter-action) so that the conflict is avoided; Second, all encountered conflicts can be checked without having to anticipate/enumerate scenarios in advance; and finally a corrective measure (such as blocking/disabling an action) is needed only when the situation demands it, not as a result of a potential conflict.

However, the detection of conflicts when they are experienced and a reactive resolution can be undesirable and sometimes dangerous in CPS-IoT systems since we effectively wait until a problem happens, rather than anticipating the problem and addressing it (if possible). There are two related approaches to this, which we term as proactive and predictive.

A. Proactive Resolution

By proactive, we mean that the system looks ahead over a specified time duration and examines the likely events based on current environmental conditions. If possible, these conflicts can be resolved proactively by modifying the operational policy associated with the involved entities. The proactive model needs to identify the events that can occur in the near future based on the environmental context, the current state of all related entities and natural processes (such as light levels or temperature), and possibly the history of previous events. Apart from higher complexity, the proactive method necessarily introduces uncertainty: the predicted/anticipated conflict may not actually occur in which case the corrective action becomes unnecessary. There is an obvious (but difficult to characterize) trade-off between how far in advance we analyze the conflicts (say, time Δt) and the corresponding false positive and false negative rates. Yet another difficulty is the characterization of likely events based on current conditions during the time Δt. If Δt is sufficiently small, it is reasonable to assume that no new events will occur, and the dynamics of the system will be governed by what is currently ongoing.

For example, if the smoke density is currently increasing due to something burning, we assume that this will continue. Similarly, if a car is getting closer to the one ahead, this will continue. Obviously, such assumptions become increasingly untenable as Δt increases, but there is also uncertainty about what actions may or may not occur as a result, and these would also need to be predicted. Furthermore, if a new event or action does occur, we need to somehow transition from the current evaluation to a new one that takes the new situation into account (e.g., start over entirely or determine what aspects of the prediction may have changed). A formal analysis of the system is essential to detect existing conflicts, anticipate potential conflicts in the future, or determine actions to resolve or avoid the conflict. The basic analysis techniques needed for this are much broader and can also be used to check other properties of the CPS-IoT system. Other than operation, this also includes access-related properties, provided that the formulation also includes allowed or disallowed accesses.

B. Predictive Resolution

By predictive, we mean a mechanism that observes the behavior of the system over time and perhaps over multiple occurrences of a similar sequence of events and actions and learns from them. The key distinction between this and proactive is that: a) predictive is not necessarily tied to short time-window behavior that we exploit in the proactive method; and b) predictive-based decisions are based on multiple occurrences of similar situations, rather than on the short-term environmental context of the situation. This would lend itself very well to the growing influence of AI algorithms in CPS-IoT systems as prediction is a task that they do very well.

To the best of our knowledge, there are no works that exploit prediction in this sense for conflict resolution purposes; however, many works attempt to predict user behavior in a CPS-IoT system. These and related mechanisms could potentially be integrated with conflict detection and resolution to provide this class of solutions. In [96], the authors proposed a framework, FORTNIoT, for making intelligible future predictions by combining self-sustaining predictions (e.g., weather forecasts) and simulations of associated ECA rules, to ascertain when these rules will trigger in the future and what state changes they will cause to connected, intelligent home entities; however, this study falls short of detecting conflicts during the prediction of future activities in a smart home.

Numerous algorithms were developed to predict the user's behavior in a smart home. For instance, a user typically awakens at 8 a.m. and immediately operates the toaster on weekends. Any method for predicting user behaviors should mine the user's behavior from such operation records and return it to the smart home control center. If tomorrow is a weekend, the system will ask the user one day in advance whether they require assistance with using the toaster at 8 a.m. In [97], the authors proposed an algorithm for Unsupervised User Behavior Prediction (UUBP) that utilizes an Artificial Neural Network (ANN) to learn user behavior along with an innovative update strategy that integrates an Ebbinghaus forgetting factor [125]. Further, a Forgetting Curve is proposed to eliminate the influence of infrequent and out-of-date operation records generated by the user. This can help to mitigate the impact of out-of-date records on the prediction process, resulting in predictive behaviors that are more consistent with recent user behaviors. This work detects and resolves the conflicts in user behavior records that might trigger the same actuator simultaneously.

In [98], the authors developed a model based on a Long Short Term Memory (LSTM) network to predict Activity of Daily Living (ADL), or the next activity that may occur after the user's current activity. [99] presented a comprehensive survey of prediction algorithms proposed in the literature for smart environments. The algorithms presented predict future events based on historical data drawn from sensors and smart devices to reduce the likelihood of malicious events occurring. This paper introduces the system models and data that are commonly used in smart prediction algorithms and discusses their features, strengths, and weaknesses in detail.

VI. Future Challenges in Conflict Detection and Resolution

Despite a significant quantity of research on the topics surveyed in this paper, numerous challenges remain. In this section, we provide a brief overview of these key challenges. First, A significant gap in previous works is represented by the inability to resolve conflicts proactively. The proposed proactive conflict detection and resolution strategy are discussed in detail in the previous section. [82], [84] considered a static analysis to check whether a collection of IoT apps working together adheres to identified safety, security, and functional properties. Further work was done in [44], [50], [89] to adapt a dynamic conflict resolution strategy by analyzing the run time behavior of an IoT application and blocking the action of an IoT app that violates defined safety and security policies.

The primary drawback of such an approach is that it may have undesirable physical consequences. For instance, suppose that a door should be unlocked only for a security service when the user is away for a vacation. A policy that disables the unlock-door state to resolve a conflict prevents the security service from entering the house, which may or may not be desirable depending on the circumstances. Much of the work that dynamically resolves conflicts (e.g., [100]) allows the user to specify policies that govern the collective behavior of the IoT applications. This presents a significant challenge in a complex IoT environment. An incorrect policy specification may prevent legitimate states, fail to block unsafe and insecure states, or conflict with another policy. However, when a proactive conflict analysis strategy is used, the operational policies or rules defined for each subsystem are checked for the occurrence of intra-subsystem conflicts.

One major issue that remains unaddressed mainly is the classification of conflicts in terms of their impact on the system's functioning. As discussed earlier, a "conflict" can be defined in many ways, and in general, it merely indicates something undesirable. The impact of the conflicts can range from minor resource inefficiency (e.g., lights being on when no one is around), to user inconvenience (e.g., temperature or luminance moderately outside the comfortable range), to detrimental (e.g., unauthorized entry allowed), to disastrous (e.g., a person trapped during a fire).

Learning such a classification automatically remains a challenge. A related issue is that some of the priorities may depend on the context rather than being static. A fundamental question that still needs more work is defining or characterizing the conflicts. As discussed earlier, many characterizations exist, and they all have pros and cons. For example, defining conflicts via (violation of) safety properties is very general. We can consider almost any type of requirement as a safety property and then enforce it using the methods discussed in this paper. Yet, it raises two questions: a) how do we come up with safety properties, and b) how do we know whether we covered all essential safety properties? Neither of these questions are well-formed, and thus cannot be answered without further specification.

Many authors characterized conflicts in terms of generic relationships between commands issued to one or more actuators. Nevertheless, a comprehensive specification of such generic situations can help identify or verify the safety properties. For example, if a specified safety parameter does not result in any of these generic situations, then it may be regarded as illegitimate. Moreover, suppose the system does enter a state that, in retrospect, is considered to be undesirable. In that case, it is helpful to consider which of these generic properties are violated and accordingly formulate a new safety parameter.

Finally, a major challenge in the field concerns the integration of AI algorithms with CPS-IoT systems. This particular problem presents a suitable area for further research, and yet as far as we have seen in the literature there is a lack of proposals that can work with the predictive algorithms methods discussed to resolve potential conflicts in CPS-IoT systems. This is in addition to the general lack of proactive methods as well. Many of the methods that we have shown in this paper mainly focus on conflict detection or formal analysis of policy conflict. While conflict resolution remains a hot area of research, it is our position that research needs to be done in integrating methods that tackle the problem of proactive and predictive conflict resolution in a large-scale, real-time, CPS-IoT system.

VII. Conclusion

In this paper, we presented a survey of works regarding operational policy conflict detection and resolution in CPS-IoT systems. We believe that, by studying this problem in a more general scope, we can illuminate the discussion of this issue in our specific context. We believe that future work in this domain should revolve more closely around integrating intelligent algorithms that take advantage of an ensemble approach. Such a method should be able to handle both static and dynamic conflicts as well as both conflicts of modality and goals. The works presented in section 4.2 go a long way toward conflict resolution in the conflict of goals case as we hypothesize that conflicts of goals can be correlated to predicted user behaviors. For example, if a user usually feels cold around 60 degrees then they are likely to want the heater turned on. This may conflict with current operational policy when the user gives a command for the heater to turn on. This can be further extended in a social context as if more people are in the room it will naturally increase the temperature. As such, more research is needed in operational policy conflict detection and resolution in order to aid the user in efficiently designing the complex operational policies that govern the operation of the CPS-IoT system while giving them the flexibility to change the rules, add new devices, or modify the system without forcing them to shut down the system to redesign or add components.

References

[1] D. W. C. Greer, M. Burns and E. Grior, "Cyber-physical systems and internet of things," Tech. Rep. NIST.SP.1900-202, National Institute for Standards and Technology, 2019.

[2] R. Baheti and H. Gill, "Cyber-physical systems," tech. rep., IEEE Control Systems Society, February 2011.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[4] Kant, Krishna and Pradeep Kumar, Pavana. (2022). Conflict Detection and Resolution in IoT Systems: A Survey. IoT. 3. 10.3390/iot3010012.

[5] J. Cano, E. Rutten, G. Delaval, Y. Benazzouz and L. Gurgen, "ECA Rules for IoT Environment: A Case Study in Safe Design," 2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops, 2014, pp. 116-121, doi: 10.1109/SASOW.2014.32.

[6] J. Hall and R. Iqbal. 2017. "CoMPES: A Command Messaging Service for IoT Policy Enforcement in a Heterogeneous Network," In Proceedings of the Second International Conference on Internet-of-Things Design and Implementation (IoTDI '17). Association for Computing Machinery, New York, NY, USA, 37–43. DOI:https://doi.org/10.1145/3054977.3054988

[7] R. Iqbal, J. Lee and J. Hall, "A Cloud Middleware Enabling Natural Speech Analysis for IoT Policy Enforcement in Smart Home Environments," 2018 IEEE International Congress on Internet of Things (ICIOT), San Francisco, CA, 2018, pp. 184-187, doi: 10.1109/ICIOT.2018.00035.

[8] R. Iqbal, J. Hall, J. H. Lee, A. Islam, "Enabling real-time audio-video inputs for Internet of Things operational policy enforcement," Internet of Things, Volume 6, 2019, 100041, ISSN 2542-6605, https://doi.org/10.1016/j.iot.2019.02.001.

[9] E. C. Lupu and M. Sloman, "Conflicts in policy-based distributed systems management," in IEEE Transactions on Software Engineering, vol. 25, no. 6, pp. 852-869, Nov.-Dec. 1999, doi: 10.1109/32.824414.

[10] M. Sloman, "Policy driven management for distributed systems," J Netw Syst Manage 2, 333–360 (1994). https://doi.org/10.1007/BF02283186

[11] Jonathan D. Moffett and Morris S. Sloman (1994) "Policy conflict analysis in distributed system management", in Journal of Organizational Computing, 4:1, 1-22, DOI: 10.1080/10919399409540214

[12] N. Dunlop, J. Indulska and K. Raymond, "Dynamic conflict detection in policy-based management systems," Proceedings. Sixth International Enterprise Distributed Object Computing, 2002, pp. 15-26, doi: 10.1109/EDOC.2002.1137693.

[13] N. Dunlop, J. Indulska and K. Raymond, "Methods for conflict resolution in policy-based management systems," Seventh IEEE International Enterprise Distributed Object Computing Conference, 2003. Proceedings., 2003, pp. 98-109, doi: 10.1109/EDOC.2003.1233841.

[14] Damianou, N., Dulay, N., Lupu, E., Sloman, M. (2001). The Ponder Policy Specification Language. In: Sloman, M., Lupu, E.C., Lobo, J. (eds) Policies for Distributed Systems and Networks. POLICY 2001. Lecture Notes in Computer Science, vol 1995. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-44569-2

[15] K. Twidle, N. Dulay, E. Lupu and M. Sloman, "Ponder2: A Policy System for Autonomous Pervasive Environments," 2009 Fifth International Conference on Autonomic and Autonomous Systems, 2009, pp. 330-335, doi: 10.1109/ICAS.2009.42.

[16] Damianou, Nicodemos and Bandara, Arosha and Sloman, Morris and Lupu, Emil. (2002). A Survey of Policy Specification Approaches.

[17] J. D. Moffett and M. S. Sloman, "Policy hierarchies for distributed systems management," in IEEE Journal on Selected Areas in Communications, vol. 11, no. 9, pp. 1404-1414, Dec. 1993, doi: 10.1109/49.257932.

[18] Lupu, E., Sloman, M. (1997). Conflict Analysis for Management Policies. In: Lazar, A.A., Saracco, R., Stadler, R. (eds) Integrated Network Management V. IM 1997. IFIP — The International Federation for Information Processing. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-35180-3

[19] Lupu, E., Sloman, M. (1997). Conflict Analysis for Management Policies. In: Lazar, A.A., Saracco, R., Stadler, R. (eds) Integrated Network Management V. IM 1997. IFIP — The International Federation for Information Processing. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-35180-3

[20] E. Lupu and M. Sloman, "Conflict Analysis for Management Policies," in Integrated Network Management V, ser. IFIP - The International Federation for Information Processing. Springer, 1997, pp. 430–443. 121

[21] D. Eppstein and S. Muthukrishnan, "Internet Packet Filter Management and Rectangle Geometry," in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '01). Society for Industrial and Applied Mathematics, 2001, pp. 827–835.

[22] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," Communications of the ACM, vol. 18, no. 9, pp. 509–517, 1975.

[23] A. Hari, S. Suri, and G. Parulkar, "Detecting and Resolving Packet Filter Conflicts," in Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000), vol. 3. IEEE, 2000, pp. 1203–1212.

[24] 24] Z. Fu, S. F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine, and C. Xu, "IPSec/VPN Security Policy: Correctness, Conflict Detection, and Resolution," in Proceedings of the International Workshop on Policies for Distributed Systems and Networks (POLICY 2001), ser. Lecture Notes in Computer Science, vol. 1995. Springer, 2001, pp. 39–56.

[25] H. Hu, G.-J. Ahn, and K. Kulkarni, "Fame: A Firewall Anomaly Management Environment," in Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration (SafeConfig '10). ACM, 2010, pp. 17–26.

[26] V. Capretta, B. Stepien, A. Felty, and S. Matwin, "Formal Correctness of Conflict Detection for Firewalls," in Proceedings of the 2007 ACM Workshop on Formal Methods in Security Engineering (FMSE '07). ACM, 2007, pp. 22–30.

[27] L. Kagal, "Rei: A Policy Language for the Me-Centric Project," HP Laboratories, Palo Alto, Technical Report, 2002.

[28] G. H. v. Wright, "Deontic Logic," Mind, vol. 60, no. 237, pp. 1–15, 1951.

[29] A. Mayer, A. Wool, and E. Ziskind, "Fang: A Firewall Analysis Engine," in Proceedings of the 2000 IEEE Symposium on Security and Privacy. IEEE, 2000, pp. 177–187.

[30] T. Tran, E. S. Al-Shaer, and R. Boutaba, "PolicyVis: Firewall Security Policy Visualization and Inspection," in LISA, vol. 7, 2007, pp. 1–16.

[31] F. Mansmann, T. Gbel, and W. Cheswick, "Visual Analysis of Complex Firewall Configurations," in Proceedings of the 9th International Symposium on Visualization for Cyber Security. ACM, 2012, pp. 1–8.

[32] C. Monsanto, J. Reich, N. Foster, J. Rexford, D. Walker, and others, "Composing Software-Defined Networks," in Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13). USENIX Association, 2013, pp. 1–13.

[33] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A Network Programming Language," in Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming (ICFP '11), vol. 46. ACM, 2011, pp. 279–291.

[34] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions Using Flowtags," in Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14). USENIX Association, 2014.

[35] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, "VeriFlow: Verifying Network-Wide Invariants in Real Time," in Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13). USENIX Association, 2013, pp. 15–27.

[36] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, "Real Time Network Policy Checking Using Header Space Analysis," in Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13). USENIX Association, 2013, pp. 99–111.

[37] P. Kazemian, G. Varghese, and N. McKeown, "Header Space Analysis: Static Checking for Networks," in Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI '12). USENIX Association, 2012, pp. 113–126.

[38] A.D. Ferguson, A. Guha, C. Liang, R. Fonseca and S. Krishnamurthi, "Hierarchical policies for software defined networks", Proceedings of the first workshop on Hot topics in software defined networks, pp. 37-42, 2012.

[39] C. N. Tran and V. Danciu, "On Conflict Handling in Software-Defined Networks," 2018 International Conference on Advanced Computing and Applications (ACOMP), 2018, pp. 50-57, doi: 10.1109/ACOMP.2018.00016.

[40] Huang, B.; Dong, H.; Bouguettaya, A. Conflict Detection in IoT-based Smart Homes. arXiv 2021, arXiv:2107.13179.

[41] Al-Farooq, A.; Al-Shaer, E.; Kant, K. A Formal Method for Detecting Rule Conflicts in Large Scale IoT Systems. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM 2019), Washington, DC, USA, 8–12 April 2019.

[42] Ma, M.; Preum, S.M.; Stankovic, J.A. Cityguard: A watchdog for safety-aware conflict detection in smart cities. In Proceedings of the Second International Conference on Internet-of-Things Design and Implementation, Pittsburgh, PA, USA, 18–21 April 2017; pp. 259–270.

[43] Liu, R.; Wang, Z.; Garcia, L.; Srivastava, M. RemedioT: Remedial actions for internet-of-things conflicts. In Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, New York, NY, USA, 13–14 November 2019 ; pp. 101–110.

[44] Celik, Z.B.; Tan, G.; McDaniel, P.D. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT; NDSS: San Diego, CA, USA, 2019.

[45] Shah, T.; Venkatesan, S.; Ngo, T.; Neelamegam, K. Conflict detection in rule based IoT systems. In Proceedings of the 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 17–19 October 2019 ; IEEE: Piscataway, NJ, USA, 2019; pp. 0276–0284.

[46] Chaki, D.; Bouguettaya, A.; Mistry, S. A Conflict Detection Framework for IoT Services in Multi-resident Smart Homes. arXiv 2020, arXiv:cs.CY/2004.12702.

[47] Abusafia, A.; Bouguettaya, A. Reliability Model for Incentive-Driven IoT Energy Services. arXiv 2021, arXiv:cs.DC/2011.06159.

[48] Lakhdari, A.; Bouguettaya, A.; Mistry, S.; Neiat, A.G.; Suleiman, B. Elastic Composition of Crowdsourced IoT Energy Services. arXiv 2020, arXiv:cs.DC/2011.06771.

[49] Huang, B.; Dong, H.; Bouguettaya, A. Conflict Detection in IoT-based Smart Homes. arXiv 2021, arXiv:2107.13179.

[50] Perumal, T.; Sulaiman, M.N.; Datta, S.K.; Ramachandran, T.; Leong, C.Y. Rule-based conflict resolution framework for Internet of Things device management in smart home environment. In Proceedings of the 2016 IEEE 5th Global Conference on Consumer Electronics, Kyoto,

Japan, 11–14 October 2016 ; IEEE: Piscataway, NJ, USA, 2016; pp. 1–2.

[51] Oh, H.; Ahn, S.; Choi, J.K.; Yang, J. Mashup service conflict detection and visualization method for Internet of Things. In Proceedings of the 2017 IEEE 6th global conference on consumer electronics (GCCE), Nara, Japan, 9–12 October 2018; IEEE: Piscataway, NJ, USA, 2017; pp. 1–2.

[52] Sun, Y.; Wang, X.; Luo, H.; Li, X. Conflict detection scheme based on formal rule model for smart building systems. IEEE Trans. Hum.-Mach. Syst. 2014, 45, 215–227.

[53] Ahmed, M.O.; Elfaki, S.E.E. Adaptation Conflicts of Heterogeneous Devices in IOT Smart-Home. Am. Acad. Sci. Res. J. Eng. Technol. Sci. 2021, 81, 64–78.

[54] Nakamura, M.; Igaki, H.; Matsumoto, K.I. Feature interactions in integrated services of networked home appliances. In Proceedings of the International Conference on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI'05) , Leicester, UK, 28–30 June 2005; pp. 236–251.

[55] Leelaprute, P.; Matsuo, T.; Tsuchiya, T.; Kikuno, T. Detecting feature interactions in home appliance networks. In Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Phuket, Thailand, 6–8 August 2008 ; IEEE: Phuket, Thailand, 2008; pp. 895–903.

[56] Leelaprute, P. Resolution of feature interactions in integrated services of home network system. In Proceedings of the 2007 Asia-Pacific Conference on Communications, Bangkok, Thailand, 18–20 October 2007 ; IEEE: Bangkok, Thailand, 2007; pp. 363–366.

[57] Igaki, H.; Nakamura, M. Modeling and detecting feature interactions among integrated services of home network systems. IEICE Trans. Inf. Syst. 2010, 93, 822–833.

[58] Li, X.; Zhang, L.; Shen, X. DIAC: An Inter-app Conflicts Detector for Open IoT Systems. ACM Trans. Embed. Comput. Syst. (TECS) 2020, 19, 1–25.

[59] Camacho, R.; Carreira, P.; Lynce, I.; Resendes, S. An ontology-based approach to conflict resolution in Home and Building Automation Systems. Expert Syst. Appl. 2014, 41, 6161–6173.

[60] Cabitza, F.; Fogli, D.; Lanzilotti, R.; Piccinno, A. Rule-based tools for the configuration of ambient intelligence systems: A comparative user study. Multimed. Tools Appl. 2017, 76, 5221–5241.

[61] TASKER For Android. Available online: https://tasker.joaoapps.com/index.html (accessed on 10 October 2021).

[62] BIPIO GRaph API. Available online: https://github.com/bipio-server/bipio/wiki (accessed on 10 October 2021).

[63] WIGWAG SMARTHOME. Available online: https://www.wigwagapp.com/ (accessed on 10 October 2021).

[64] ZIPATILE2. Available online: https://www.zipato.com/ (accessed on 10 October 2021).

[65] Sun, Q.; Yu, W.; Kochurov, N.; Hao, Q.; Hu, F. A multi-agent-based intelligent sensor and actuator network design for smart house and home automation. J. Sens. Actuator Netw. 2013, 2, 557–588.

[66] Luo, H.; Wang, R.; Li, X. A rule verification and resolution framework in smart building system. In Proceedings of the 2013 International Conference on Parallel and Distributed Systems, Seoul, Korea, 15–18 December 2013; IEEE: Seoul, Korea, 2013; pp. 438–439.

[67] Maternaghan, C.; Turner, K.J. Policy conflicts in home automation. Comput. Netw. 2013, 57, 2429–2441.

[68] Newcomb, J.L.; Chandra, S.; Jeannin, J.B.; Schlesinger, C.; Sridharan, M. IOTA: A calculus for internet of things automation. In Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Vancouver, BC, Canada, 25–27 October 2017 ; pp. 119–133.

[69] . Bak, N.; Chang, B.M.; Choi, K. Smart block: A visual programming environment for smartthings. In Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 23–27 July 2018; Volume 2, pp. 32–37.

[70] Sen, J. Internet of Things: Technology, Applications and Standardization; BoD–Books on Demand: Norderstedt, Germany, 2018.

[71] Xie,W.; Zhu, H.;Wu, X.; Vinh, P.C. Formal verification of mCWQ using extended Hoare logic. Mob. Netw. Appl. 2019, 24, 134–144.

[72] Shehata, M.; Eberlein, A.; Fapojuwo, A. Using semi-formal methods for detecting interactions among smart homes policies. Sci. Comput. Program. 2007, 67, 125–161.

[73] Shehata, M.; Eberlein, A.; Fapojuwo, A.O. A taxonomy for identifying requirement interactions in software systems. Comput. Netw. 2007, 51, 398–425.

[74] Carreira, P.; Resendes, S.; Santos, A.C. Towards automatic conflict detection in home and building automation systems. Pervasive Mob. Comput. 2014, 12, 37–57.

[75] Android Things Website. Available online: https://developer.android.com/things (accessed on 10 October 2021).

[76] SAMSUNG Smartthings. Available online: https://www.samsung.com (accessed on 10 October 2021).

[77] Apple Homekit 2021. Available online: https://www.apple.com/shop/accessories/all/homekit (accessed on 10 October 2021).

[78] OpenHAB 2021. Available online: https://www.openhab.org/ (accessed on 10 October 2021).

[79] Shen, X.; Zhang, L.; Li, X. A Systematic Examination of Inter-App Conflicts Detections in Open IoT Systems; Technical Report; North Carolina State University, Department of Computer Science: Raleigh, NC, USA, 2017.

[80] Liang, C.J.M.; Karlsson, B.F.; Lane, N.D.; Zhao, F.; Zhang, J.; Pan, Z.; Li, Z.; Yu, Y. SIFT: Building an internet of safe things. In Proceedings of the 14th International Conference on Information Processing in Sensor Networks, Seattle, WA, USA, 14–16 April 2015; pp. 298–309.

[81] Trimananda, R.; Aqajari, S.A.H.; Chuang, J.; Demsky, B.; Xu, G.H.; Lu, S. Understanding and automatically detecting conflicting interactions between smart home IOT applications. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Online, 8–13 November 2020; pp. 1215–1227.

[82] Celik, Z.B.; McDaniel, P.; Tan, G. Soteria: Automated IOT safety and security analysis. In Proceedings of the 2018 USENIX ATC, Boston, MA, USA, 11–13 July 2018; pp. 147–158.

[83] Munir, S.; Stankovic, J.A. Depsys: Dependency aware integration of cyber-physical systems for smart homes. In Proceedings of the 2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS), Berlin, Germany, 14–17 April 2014; pp. 127–138.

[84] Nguyen, D.T.; Song, C.; Qian, Z.; Krishnamurthy, S.V.; Colbert, E.J.; McDaniel, P. IotSan: Fortifying the safety of IoT systems. In Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies, Heraklion, Greece, 4–7 December 2018; pp. 191–203.

[85] Ding, W.; Hu, H. On the safety of IOT device physical interaction control. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 832–846.

[86] Ma, M.; Stankovic, J.A.; Feng, L. Cityresolver: A decision support system for conflict resolution in smart cities. In Proceedings of the 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), Porto, Portugal, 11–13 April 2018; IEEE: Porto, Portugal, 2018; pp. 55–64.

[87] Ma, M.; Stankovic, J.A.; Feng, L. Runtime monitoring of safety and performance requirements in smart cities. In Proceedings of the 1st ACMWorkshop on the Internet of Safe Things, Delft, The Netherlands, 5 November 2017; pp. 44–50.

[88] Ma, M.; Preum, S.M.; Stankovic, J.A. Demo abstract: Simulating conflict detection in heterogeneous services of a smart city. In Proceedings of the 2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI), Pittsburgh, PA, USA, 18–21 April 2017; pp. 275–276.

[89] Chaki, D.; Bouguettaya, A. Adaptive priority-based conflict resolution of IoT services. In Proceedings of the 2021 IEEE International Conference onWeb Services (ICWS), Chicago, IL, USA, 5–10 September 2021; pp. 663–668.

[90] Berners-Lee, T.; Hendler, J.; Lassila, O. The semantic web. Sci. Am. 2001, 284, 34–43. IoT 2022, 3 217

[91] Grau, B.C.; Horrocks, I.; Motik, B.; Parsia, B.; Patel-Schneider, P.; Sattler, U. OWL 2: The next step for OWL. J. Web Semant. 2008, 6, 309–322.

[92] Bouquet, P.; Giunchiglia, F.; Van Harmelen, F.; Serafini, L.; Stuckenschmidt, H. Contextualizing ontologies. J. Web Semant. 2004, 1, 325–343.

[93] Horrocks, I.; Patel-Schneider, P.F.; Boley, H.; Tabet, S.; Grosof, B.; Dean, M. SWRL: A semantic web rule language combining OWL and RuleML. W3C Memb. Submiss. 2004, 21, 1–31.

[94] Celik, Z.B.; Babun, L.; Sikder, A.K.; Aksu, H.; Tan, G.; McDaniel, P.; Uluagac, A.S. Sensitive information tracking in commodity IoT. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 1687–1704.

[95] Ma, M.; Preum, S.M.; Tarneberg, W.; Ahmed, M.; Ruiters, M.; Stankovic, J. Detection of runtime conflicts among services in smart cities. In Proceedings of the 2016 IEEE International Conference on Smart Computing (SMARTCOMP), St. Louis, MO, USA, 18–20 May 2016 ; IEEE: Piscataway, NJ, USA, 2016; pp. 1–10.

[96] Coppers, S.; Vanacken, D.; Luyten, K. FORTNIoT: Intelligible Predictions to Improve User Understanding of Smart Home Behavior. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2020, 4, 1–24.

[97] Liang, T.; Zeng, B.; Liu, J.; Ye, L.; Zou, C. An unsupervised user behavior prediction algorithm based on machine learning and neural network for smart home. IEEE Access 2018, 6, 49237–49247.

[98] Du, Y.; Lim, Y.; Tan, Y. A novel human activity recognition and prediction in smart home based on interaction. Sensors 2019, 19, 4474.

[99] Wu, S.; Rendall, J.B.; Smith, M.J.; Zhu, S.; Xu, J.; Wang, H.; Yang, Q.; Qin, P. Survey on prediction algorithms in smart homes. IEEE Internet Things J. 2017, 4, 636–644.

[100] Nagendra, V.; Bhattacharya, A.; Yegneswaran, V.; Rahmati, A.; Das, S.R. VISCR: intuitive and conflict-free automation for securing the dynamic consumer IOT infrastructures. arXiv 2019, arXiv:1907.13288.

[101] Chi, H.; Zeng, Q.; Du, X.; Yu, J. Cross-app interference threats in smart homes: Categorization, detection and handling. In Proceedings of the 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Valencia, Spain, 29 June–2 July 2020 ; IEEE: Piscataway, NJ, USA, 2020; pp. 411–423.

[102] Fernandes, E.; Paupore, J.; Rahmati, A.; Simionato, D.; Conti, M.; Prakash, A. Flowfence: Practical data protection for emerging iot application frameworks. In Proceedings of the 25th USENIX security symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 531–548.

[103] Bastys, I.; Balliu, M.; Sabelfeld, A. If this then what? Controlling flows in IoT apps. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1102–1119.

[104] Mohsin, M.; Anwar, Z.; Husari, G.; Al-Shaer, E.; Rahman, M.A. IoTSAT: A formal framework for security analysis of the Internet of Things. In Proceedings of the IEEE Conference on Communications and Network Security (CNS), Philadelphia, PA, USA, 17–19 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–7.

[105] Tian, Y.; Zhang, N.; Lin, Y.H.;Wang, X.; Ur, B.; Guo, X.; Tague, P. Smartauth: User-centered authorization for the internet of things. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017; pp. 361–378.

[106] Zhang,W.; Meng, Y.; Liu, Y.; Zhang, X.; Zhang, Y.; Zhu, H. Homonit: Monitoring smart home apps from encrypted traffic. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1074–1088.

[107] Pandita, R.; Xiao, X.; Yang, W.; Enck, W.; Xie, T. WHYPER: Towards automating risk assessment of mobile applications. In Proceedings of the 22nd USENIX Security Symposium (USENIX Security 13), Washington, DC, USA, 14–16 August 2013; pp. 527–542.

[108] Wang, J.; Hao, S.; Wen, R.; Zhang, B.; Zhang, L.; Hu, H.; Lu, R. IoT-praetor: Undesired behaviors detection for IoT devices. IEEE Internet Things J. 2020, 8, 927–940.

[109] Jia, Y.J.; Chen, Q.A.;Wang, S.; Rahmati, A.; Fernandes, E.; Mao, Z.M.; Prakash, A.; Unviersity, S. ContexloT: Towards Providing Contextual Integrity to Appified IoT Platforms. In Proceedings of the NDSS, San Diego, CA, USA, 26 February–1 March 2017; p. 2.

[110] Wang, Q.; Hassan, W.U.; Bates, A.; Gunter, C. Fear and logging in the internet of things. In Proceedings of the Network and Distributed Systems Symposium, San Diego, CA, USA, 18–21 February 2018.

[111] Babun, L.; Celik, Z.B.; McDaniel, P.; Uluagac, A.S. Real-time analysis of privacy-(un) aware IoT applications. arXiv 2019, arXiv:1911.10461.

[112] Ernst, M.D. Static and Dynamic Analysis: Synergy and Duality; MIT Computer Science and Artificial Intelligence Lab: Cambridge, MA, USA, 2003.

[113] de Moura, L.; Bjørner, N. Z3: An Efficient SMT Solver. In TACAS; Ramakrishnan, C.R., Rehof, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 337–340.

[114] Cavada, R.; Cimatti, A.; Dorigatti, M.; Griggio, A.; Mariotti, A.; Micheli, A.; Mover, S.; Roveri, M.; Tonetta, S. The nuXmv symbolic model checker. In Proceedings of the 26th International Conference on Computer Aided Verification, Vienna, Austria, 18–22 July 2014; pp. 334–342. IoT 2022, 3 218

[115] Ansótegui, C.; Bonet, M.L.; Levy, J. A new algorithm for weighted partial MaxSAT. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 1 December 2009–18 January 2010 .

[116] Hoos, H.; Sttzle, T. Stochastic Local Search: Foundations and Applications; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2004.

[117] Bianchi, L.; Dorigo, M.; Gambardella, L.M.; Gutjahr, W.J. A survey on metaheuristics for stochastic combinatorial optimization. Nat. Comput. 2009, 8, 239–287.

[118] Tolson, B.; Shoemaker, C. Dynamically dimensioned search algorithm for computationally efficient watershed model calibration. Water Resour. Res. 2007, 43.

[119] Arsenault, R.; Poulin, A.; Côté, P.; Brissette, F. Comparison of stochastic optimization algorithms in hydrological model calibration. J. Hydrol. Eng. 2014, 19, 1374–1384. [CrossRef]

[120] Mohammadi, N.; Sondur, S.; Kant, K. Effective Configuration Optimization of Large Scale Software Systems. 2022.

[121] Mezura-Montes, E.; Coello, C. Constraint-handling in nature-inspired numerical optimization: Past, present and future. Swarm Evol. Comput. 2011, 1, 173–194.

[122] Riera, B.; Emprin, F.; Annebicque, D.; Colas, M.; Vigario, B. HOME I/O: A virtual house for control and STEM education from middle schools to Universities. IFAC-Papers OnLine 2016, 49, 168–173.

[123] Pradeep, P.; Kant, K.; Pal, A. Managing Access Control in Large-Scale Multi-Party IoT Systems. In Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid 2022), Taormina, Sicily, Italy, 16–19 May 2022.

[124] Ouaddah, A.; Abou Elkalam, A.; Ouahman, A.A. Towards a novel privacy-preserving access control model based on blockchain technology in IoT. In Europe and MENA Cooperation Advances in Information and Communication Technologies; Springer: Cham, Switzerland, 2017; pp. 523–533.

[125] Ebbinghaus, Hermann (1913). Memory: A Contribution to Experimental Psychology. Translated by Ruger, Henry; Bussenius, Clara. New York city, Teachers college, Columbia university.