Management Application Interactions in Software-based Networks

Daphne Tuncer, Marinos Charalambides, and George Pavlou

***Abstract*—To support the next wave of networking technologies and services, which will likely involve heterogeneous resources and requirements, rich management functionality will need to be deployed. This raises questions regarding the interoperability of such functionality in an environment where potentially interacting applications operate in parallel. Interactions can cause configuration instabilities and subsequently network performance degradation, especially in the presence of contradicting objectives. Detecting and handling these interactions is therefore essential. In this paper we present an overview of the interaction management problem, a critical issue in software-based networks. We review and compare existing solutions proposed in the literature and discuss key challenges towards the development of a generic framework for the automated and real-time management of these interactions.**

I. Introduction

Today's communication networks are highly complex infrastructures that are challenging to manage. On one hand, they support a wide range of services with various characteristics, from basic connectivity to ultra high-definition online video streaming or time-critical applications for cyber-physical systems. On the other, they require the configuration of heterogeneous resources that not only include links and routers but also storage [1] and computation [2]. In addition, to operate the infrastructure and the services running over it, these systems call for rich network management functionality. Reducing the complexity associated with network management is thus crucial.

The emergence of novel networking models in the recent years, with initiatives such as software-defined networking (SDN), as well as promising advances in the domain of network programmability (*e.g.,* P4 [1]) and abstraction, have opened up new perspectives for the development of solutions that simplify management tasks. In particular, through dynamic adaptation of the forwarding plane configuration, SDN coupled with network programmability has the potential to offer flexibility in managing the network resources at run-time. In addition, the development of new abstractions facilitates the deployment of more complex management functionality by exposing a simplified view of the infrastructure to the management system.

Daphne Tuncer (corresponding author) is with the Department of Computing, Imperial College London, London SW7 2AZ, UK (e-mail: dtuncer@ic.ac.uk). Marinos Charalambides, and George Pavlou are with the Department of Electronic and Electrical Engineering, University College London, London WC1E 7JE, UK (e-mail: marinos.charalambides@ucl.ac.uk; g.pavlou.ucl.ac.uk)

[1]https://p4.org/ [accessed-30-Jan-2019]

In software-based networks the management functionality is executed through the management layer that hosts a set of *management applications* (MAs) implemented as independent modules, each responsible for a specific resource configuration function. Modularity is a key feature of the well-known SDN architecture and mainly applies to decision-making functions and control plane implementations. By enabling a clear separation of concerns between different functionalities, not only does it simplify the integration/update/removal of the components of the system, it also allows different vendor implementations to be used together and facilitates extensibility.

Despite these significant advantages, the use of modular structures for the implementation of MAs raises questions regarding their interoperability. Due to potential overlaps in the resources being managed, incompatible objectives or synchronization issues, *interactions* can occur between seemingly independent applications operating in the same environment. Although interactions can be harmless under some conditions, in other cases their existence constitutes a critical problem given that they can have an adverse impact on the operation of the network. In fact, interactions can not only lead to performance degradation, they can also be the source of instability, which is a primary concern for network operators. Detecting and handling interactions is therefore essential.

Current practices for the management of interactions mostly rely on manual and static solutions that, in addition to being challenging to implement, are also not well-suited for the requirements of flexible, adaptive and extensible management systems that software-based networks aim to support. Mechanisms that enable interactions to be detected and mitigated in an automated way and in real-time are instead needed. Developing such mechanisms is however not trivial given the wide range of applications (*e.g.,* traffic engineering, security management, server selection), their implementation complexity, and the time constraints within which their operation should be harmonized.

In this paper we aim at drawing the attention on a critical issue concerning the management of software-based networks by studying the interaction problem between MAs. Our work lies within the research context addressing modular programming in SDN, *e.g.,* [3] [4], complementing these efforts by specifically focusing on the management layer. To this end, we review the state-of-the art, discuss and classify existing mitigation solutions, and present a qualitative evaluation of various strategies. Based on this analysis, we propose a model for the implementation of the interaction management functionality that enables the best mitigation strategy to be automatically selected given

TABLE I
MANAGEMENT APPLICATION CATEGORIES.

Category	Responsibility	Examples
Device management	Control the configuration of physical devices in the managed infrastructure	Hardware power control Firmware upgrade Device configuration setup
Commodity management	Control the usage of available commodities (*i.e.,* bandwidth, CPU, memory)	Path management Virtual machine placement/migration Job scheduling Flow table usage optimization Content placement
Traffic management	Decisions on the treatment of network traffic	Traffic engineering Request redirection (server selection) Traffic analysis

the types of detected interactions between MAs and current operating conditions. We further identify key challenges to be addressed towards the development of a generic framework for the automated and real-time management of interactions. We believe that solving these challenges is essential before software-based networks can be deployed in the real.

II. MANAGEMENT APPLICATIONS AND THEIR INTERACTIONS

A. Management Applications

The sequence of operations to manage resources is commonly represented as a control-loop process involving two key functions: monitoring and decision-making. The monitoring functionality is responsible for collecting raw statistics from the resources and extracting information regarding the current conditions in the underlying infrastructure. This information is provided as input to Management Applications (MAs) that implement the decision-making logic to configure the resources according to some high-level objectives (*e.g.,* to reduce energy consumption). Based on the received input, each MA analyzes the context in which it is operating, decides on the adjustments to perform and computes new configurations, which are finally enforced on the underlying infrastructure. Existing MAs can be classified in three main categories as shown in Table I.

Each MA is represented in the management system as a software module that can be written in any programming language. MAs have different characteristics in terms of the type of resources they manage (*i.e.,* configuration parameters), the timescale at which they operate and the execution scheme they follow (proactive *vs.* reactive). They have also various requirements with respect to the scope and granularity of the information they need to make a decision. To decide on the reconfigurations, MAs take into account values of long-lived parameters, such as infrastructure static attributes (*e.g.,* link/storage/processing capacity *etc.*), as well as short-lived parameters, representing the context dynamics (*e.g.,* resource utilization, demand *etc.*). These characteristics directly drive the way in which MAs should be implemented (centralized *vs.* decentralized) in the management system. They also influence the level of abstraction required to implement each MA's functionality and to represent the resources on which they operate.

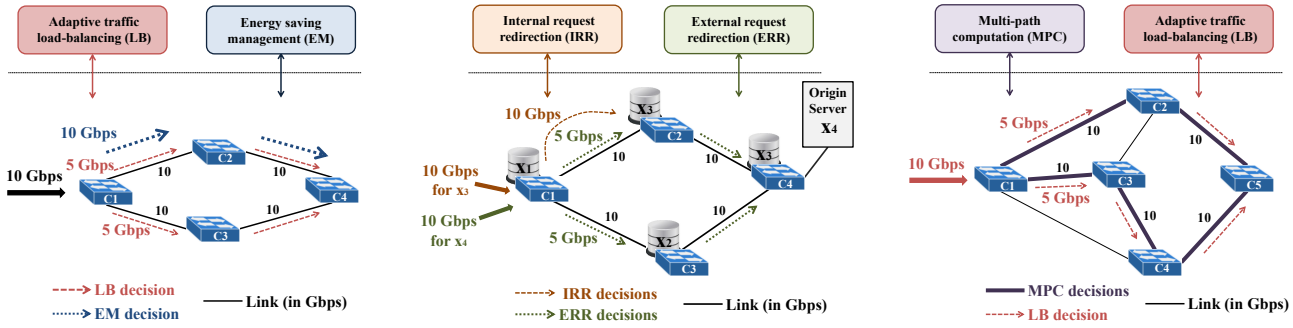
B. Management Application Interactions

From a formal point of view, interactions occur between independent applications when commonalities exist in the set of parameters each one operates on. We can distinguish three scenarios under which interactions occur between pairs of MAs operating in the same environment.

1) Shared Access to Configuration Parameters: This concerns interactions arising when two MAs are granted shared access to the same set of configuration parameters. This type of interaction can be classified as a *conflict* since the decisions taken by one application are overwritten by the other. The MAs do not need to operate at the same timescale or simultaneously for the interactions to manifest. They occur when at least one of the MAs can modify the value of the configuration parameters computed by the other, which are to be applied over the next configuration period of that MA.

Adaptive traffic load-balancing (LB) and energy saving (EM) applications are illustrative examples of cases where such an interaction can arise, as depicted in Fig. 1a. Here, each application needs to decide how to split the 10 Gbps of traffic between nodes $C1$ and $C4$ in order to satisfy its own optimization objective. To minimize link utilization, LB decides to split it in equal proportions, *i.e.,* 5 Gbps on each path. In contrast, EM concentrates all traffic on one path to minimize the number of active links. As a result, the decisions taken by each MA individually lead to inconsistent network configurations.

2) Competitive Access to Shared Resources: This concerns interactions arising when applications, while controlling different configuration parameters, affect the same resources. While this type of interaction can be harmless in situations with ample resources, it requires special attention when the shared



(a) Shared Access to Configuration Parameters (b) Competitive Access to Shared Resources (c) Functional Dependencies

Fig. 1. Examples of interaction scenarios.

resources are limited as it constitutes, in that case, a resource contention problem.

A representative example is the case investigated in [5] which focuses on the interaction between a traffic engineering application and a multimedia content server selection application. These MAs are particularly relevant in the context of an Internet Service Provider (ISP)-operated content delivery service where an ISP deploys caching points within its network to store popular content items [1]. Here, an internal request redirection (IRR) application determines the caching point to which internally served requests should be sent, while an external request redirection (ERR) application selects the route for requests redirected to the origin server. The two applications are directly coupled given that their decisions affect the same resources, *e.g.,* in terms of link utilization. An example of a conflicting decision is depicted in Fig. 1b. IRR decides to redirect the requests for content x_3 to the closest caching location C_2. At the same time, ERR decides to equally balance the requests for x_4 between the two paths to the origin server. Due to uncoordinated decisions, the traffic volume sent over C_1-C_2 exceeds the link capacity, resulting in packet losses and overall performance degradation.

3) Functional Dependencies: This scenario concerns interactions resulting from the existence of functional dependencies between two MAs. In particular, applications interacting based on this type are representative of the case where the execution of one MA is conditional on the execution of the other, *i.e.,* the input of one MA requires the output of the other MA to operate. In other words, the MAs need to be composed to manage the resources.

Interacting MAs, in this case, do not necessarily compete for shared resources. An example is the composition between a multi-path computation (MPC) application and an adaptive LB application as illustrated in Fig. 1c. The paths to the destination should be computed before LB can decide on the volume of traffic across each path. The decisions taken by LB have, however, no impact on the configurations applied by MPC.

Applications with functional dependencies that compete for access over the same resources are in general strongly coupled. As such, it is very likely that both are implemented in the management system. A typical example is a content placement application responsible for selecting a subset of content items to store in the network and the IRR application presented in Section II-B2, both aiming at reducing network bandwidth usage while controlling different configuration parameters.

An overview of the possible relationships between pairs of MAs is depicted in Fig. 2. The case where MAs have no functional dependencies and do not compete for access to shared resources represents independent non-interacting applications (type 1.2). Given the nested relationship between some applications, the type of interaction is defined by the most specific case, *e.g.,* types 2.1.1 and 2.2.1 where applications do not only compete for shared resources but also for configuration parameters.

III. Detection and Mitigation

Managing interactions involves two operations: *i) detection* to determine whether interactions can occur between applications and if so, identify the interaction type, and *ii) mitigation* to compute mitigating actions and handle the detected interactions. Various approaches have been developed in different contexts to deal with the detection and resolution of inconsistencies and/or conflicts that may arise due to multiple decision-making processes operating in the same environment, *e.g.,* [4] [6]. We review the main solutions found in the literature with more emphasis on mitigation since the resulting decisions directly affect the configuration of network resources.

A. Detection

The main principle of mechanisms developed for the detection of inconsistent configurations is to analyze dependencies between interacting components. Detection approaches have received particular attention in the area of policy-based management, in which conflicts can arise as a result of contradictory policy-driven operations simultaneously enforced on the managed system. Various conflict types have been identified in the literature, which have been broadly classified into domain-independent and domain-specific. The former are independent

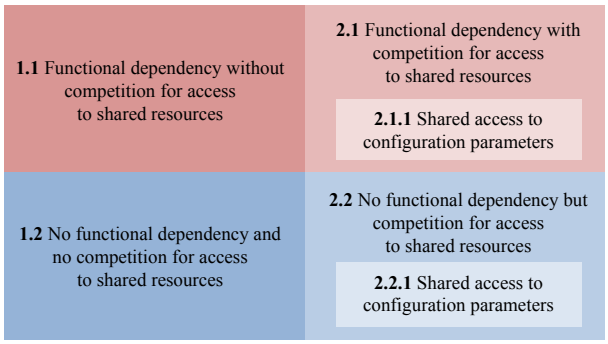


Fig. 2. Overview of possible relationships between management applications.

of the policy domain, whereas the latter are bound by the constraints of the application domain. Most detection approaches are based on the definition of the conditions under which a conflict would arise and the evaluation of those conditions during a detection process. Although domain-independent conflicts can be detected by simple syntactic analysis, more specialized inconsistencies require additional information, which can include domain- and system-specific knowledge. As reported in [6], the main detection methods are based on meta-policies, rule relations, applicability spaces, and information models, each having its own merits.

Dependencies in the context of interactions between MAs are determined from the commonalities that exist between MAs in terms of configuration parameters and/or shared resources. Detection approaches work in two steps. They are first responsible for identifying the set of MAs that either modify or access a common set of environment variables or parameters. This is achieved with models to represent the environment, *e.g.,* [7], and mechanisms to derive for each application a set of descriptors characterizing the information it needs to take decisions and operate, *e.g.,* [8] [9].

Based on the cartography of such dependencies, the second step of detection approaches consists in checking for potential inconsistencies/conflicts and invariants violations (*e.g.,* exceed capacity constraints), which can either be realized offline based on static analysis of possible interactions or online. The choice of the method to use mainly depends on whether the current state of the managed system needs to be taken into account to identify and characterize the interactions. In particular, static analysis can be performed each time a MA is newly introduced in the system or modified, whereas dynamic analysis should be executed each time a new decision is computed.

B. Mitigation

Techniques to mitigate interactions can be grouped in four categories: *i)* avoidance, *ii)* ordering, *iii)* precedence and *iv)* harmonization.

1) Avoidance-based Strategies: Their objective is to *avert* the occurrence of interactions between applications. Proposed solutions revolve around two approaches:

- **Resource Partitioning** that aims at isolating MAs from each other by allocating a dedicated slice of the available resources on which they have exclusive access. This prevents interactions to occur given that the sets of parameters associated with each application are by design disjoint.
- **Single Optimization** that replaces individual applications with a single one to avoid configuration decisions being taken independently by different MAs. The principle of these strategies is to model the resulting application as a global optimization problem, using for instance Multi-Objective Optimization (MOO), and define the global configuration objective as a combination of each individual objective (*e.g.,* weighted sum).

2) Ordering-based Strategies: Their objective is to determine the order according to which applications should be executed. Given that two MAs can access the network resources and change their configuration parameters at different instants in time, a traditional approach is to use a **Temporal Decomposition** to order the applications. This solution is particularly well suited when the applications operate at different timescales. In this case, the shorter timescale application operates within the constraints set by the longer timescale one.

3) Precedence-based Strategies: A popular approach for mitigating interactions is to establish precedence between applications. The objective is to assign MAs with priorities in order to determine which one should prevail in the event of inconsistent decisions. Two rationals can be followed to decide on the priorities.

- **Time-based** The choice of the prevailing application is based on the time at which the MAs' configuration decisions are received. A typical example is the *last-writer win* strategy used in [7] that selects the decisions of the MAs with the most recent new configuration.
- **Condition-based** Priorities are computed so that the prevailing MA is the one whose execution is concomitant to the occurrence of some network events/conditions.

4) Harmonization-based Strategies: Their objective is to harmonize the decisions taken by each MA so that individual objectives can be satisfied. The goal is not only to ensure that the resulting configurations can guarantee the overall system consistency but also that acceptable levels of performance are maintained for each application.

- **MOO-based** A trade-off is applied between the objectives of each MA by either considering their weighted sum or through the introduction of priorities. In this case, the resulting configuration is such that optimal performance is achieved for the highest priority objective, whereas only acceptable performance levels can be guaranteed for the rest. In contrast to Single Optimization, configuration decisions are taken individually by each MA.
- **Game-theory based** The joint optimization problem is modeled as a game between multiple players where each player, representing an application, tries to improve the value of its own objective.

TABLE II
MITIGATION STRATEGY COMPARISON.

Technique	Strategy	Extensibility	Scalability	Configurability	Fairness	Complexity
Avoidance	Resource Partitioning	-	-	-	-	O
	Single Optimization	-	-	-	+	-
Ordering	Temporal Decomposition	+	+	+	-	+
Precedence	Time-based	+	+	+	-	+
	Condition-based	+	+	+	-	+
Harmonization	MOO-based	O	O	+	+	O
	Game-theory-based	O	O	+	+	O

C. Mitigation Strategy Comparison

The execution time and overhead to compute remedy actions is an important factor for the network resource management system. An efficient solution for the mitigation of interactions is expected to satisfy five desirable properties:

- Extensibility: The impact of the addition/removal/modification of MAs on the mitigation strategy should be minimal. The smaller the impact, the more extensible the strategy.
- Scalability: The time needed by the strategy to compute mitigation actions should at most increase proportionally with the number of interacting MAs. The less the increase in time, the more scalable the strategy.
- Configurability: The task of changing the logic by which interactions are handled (*e.g.,* changing priorities) should be performed through simple parameter tuning. The easier the task, the higher the configurability. The degree of configurability of a strategy accounts both for its flexibility and adaptability.
- Fairness: The mitigation solution computed by the strategy should be impartial to the requirements of the interacting MAs. The less partial the solution, the fairer the strategy.
- Complexity: The cost for the strategy to compute a solution should be minimal. The lower the cost, the less complex the strategy.

In Table II we compare the different strategies presented in Section III-B by assessing their quality with respect to the aforementioned properties. For each strategy, we mark as (+) the properties that have a positive attribute and as (-) the ones that have a negative attribute. The symbol (o) is used when strategies are neutral with respect to a property.

Avoidance-based strategies do not perform well and should only be used in cases where it is possible to eliminate interactions at the design phase of the applications, *e.g.,* strongly coupled applications developed at the same time, or applications mapped to separate parts of the infrastructure. Ordering and precedence-based strategies offer in general good performance. In addition to their relative simplicity in terms of implementation, they come with a high degree of extensibility, scalability and configurability. They do however fail to provide fairness guarantees and may therefore lead to sub-optimal resource usage. In contrast, harmonization techniques offer strong guarantees in terms of fairness. However, the configuration of the mitigation mechanisms derived from these techniques needs to reflect the specifics of the concurrent applications (*e.g.,* comparative priorities, individual best/worst performance *etc.*), which is not always straightforward to achieve in practice. They also have a lower degree of extensibility and scalability since handling interactions requires coordinating the decisions of individual MAs.

Selecting *a priori* which mitigation strategy to use is not trivial given that they all have different pros and cons. In practice, the choice of a strategy should not only depend on the type of interaction to handle but also on the static and dynamic characteristics of the environment in which these arise. From a system perspective, this means that, in addition to computing mitigating actions, the mitigation function should also be responsible for automatically selecting which strategy to use based on current operating conditions.

IV. INTERACTION MANAGEMENT FUNCTIONALITY

In addition to the MAs' characteristics, the effects of interactions also depend on the environment in which these operate (*e.g.,* resource availability and operating conditions). To efficiently handle interactions, we propose to complement the network management process with an interaction management functionality that operates in three steps as depicted in Fig. 3. The objective is to supervise the operations of coexisting applications by 1) ensuring that independent MAs can gracefully operate in parallel, and 2) facilitating the integration/update/removal of MAs from the operating environment without causing disruption.

The proposed process involves three main functions: detection, selection and mitigation. To preserve the modularity of the management system, these are implemented as separate components. More specifically, the detection component uses information exposed by each MA regarding the set of parameters it operates on to identify possible interactions based on

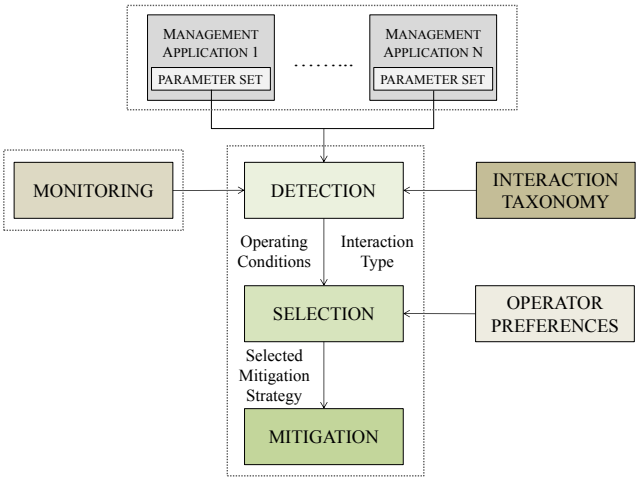


Fig. 3. Interaction management process.

the taxonomy presented in Section II-B. In addition it uses information provided by the monitoring functionality regarding the operating conditions to classify the current state of the underlying infrastructure as either *normal* or *critical*. The results are passed to the selection component that decides which strategy to execute to mitigate the detected interaction given the performance attributes defined in Section III-C. The selection is carried out in two steps. The procedure first determines which technique best applies based on the interaction type and network state. It then fine-tunes the selection by choosing a specific strategy (in case of multiple options) following preferences set by the operator. The mitigation component finally triggers the relevant mitigating actions.

The selection rules are presented in Algorithm 1. In case of functional dependencies, ordering is always the recommended technique. For the other interaction types, the best choice depends on the network state. In particular, the procedure follows a trade-off between *optimization* and *responsiveness*. While under normal operating conditions time can be devoted to compute configurations that optimize all the objectives, *i.e.,* harmonizing the decisions, the focus is on making sure that the system remains operational under critical conditions, *i.e.,* give precedence to imperative applications.

Algorithm 1 Mitigation strategy selection rules.

Inputs: InteractionType; OperatingConditions.
if InteractionType==SharedAccessToConfigurationParameters **then**
  if OperatingConditions==Normal **then**
    ORDERING
  end if
  if OperatingConditions==Critical **then**
    PRECEDENCE
  end if
end if
if InteractionType==CompetitiveAccessToSharedResources **then**
  if OperatingConditions==Normal **then**
    HARMONIZATION
  end if
  if OperatingConditions==Critical **then**
    PRECEDENCE
  end if
end if
if InteractionType==FunctionalDependencies **then**
  ORDERING
end if
Output: SelectedMitigationStrategy.

V. Open Challenges

The issue of interactions between concurrent management applications is not new to the research community that has invested efforts in understanding how interactions can arise and in proposing strategies to handle uncoordinated decisions. Today's solutions for the management of interactions heavily rely on *ad hoc* practices that usually involve the intervention of a human operator and whereby the set of actions to execute when interactions arise follows some static pre-determined rules. These practices are not well suited for emerging software-based networks, which are more complex and richer in functionality. To efficiently handle interactions in these environments, solutions that allow for a fully automated interaction management functionality operating in real-time are needed. While some proposals with a partial degree of automation and adaptability have been presented in the recent years [7] [10], we believe that more research is needed to address this challenging problem. In this section we discuss three important open issues.

A. Models for Network Resource Management

A key issue to investigate concerns the availability of abstraction models to represent management applications. Over the years, there have been some efforts in the development of standardized modeling frameworks for network resources and configurations, such as YANG[2] or SID[3]. With the advent of new paradigms such as SDN and NFV, the need for common abstraction models has been pushed in the limelight as a key critical issue [11]. Defining the right abstraction is crucial to enable interoperability between applications and understand their interdependence, as well as their relation to the environment.

A number of initiatives[4] in recent years have been investigating novel programming frameworks with the objective of providing network operators with an interface for specifying the functionality of their network at a high-level of abstraction. Until now however, the effort invested towards the development of abstractions for network management applications has been limited. In [12] a high-level abstraction in the form of path-based optimization is proposed to represent MAs, while in [13] MAs are modeled as sets of atomic stateful functions interacting

[2]IETF RFC 6020, October 2010
[3]https://www.tmforum.org/information-framework-sid/
[4]http://frenetic-lang.org/

between themselves by exchanging asynchronous messages. In [8] each application is described as a view on a SQL database that represents the whole network control infrastructure. While these abstractions were developed for different purposes they share a common objective, which is to provide means to reason upon management functionality in a general way. This is a significant step towards the development of a framework to enable the systematic analysis of management applications and their interactions. In that direction we believe that further efforts are needed to investigate which formal semantics and language paradigms are best suited to express network resource management functionality, hence complementing proposals focusing on the control functionality. These can cater for effective treatment of interactions as they can provide *(i)* vital information for their detection, and *(ii)* precise explanations for their occurrence that can be used to select the most appropriate mitigation strategy and associated configuration (cf. Section IV).

B. Self-Adaptive Mitigation

Given that the effects of interactions depend on the operating conditions, a key challenge is to implement mitigation functions that can automatically infer, in an online fashion, the best set of mitigating actions to enforce based on current operating conditions (*e.g.,* traffic dynamics, interacting MAs *etc.*). This involves the development of mechanisms that can adapt at run-time the parameters of mitigation strategies, as well as approaches that can automatically select from a range of possible strategies the one to use for achieving a specific objective.

An interesting avenue of research is to investigate the use of learning techniques to design such self-adaptive mitigation approaches. The application of machine learning as a driver for the realization of self-adaptive management functionality received a lot of attention in the context of the research on autonomic networking [14], in particular techniques based on reinforcement learning. A relevant example for self-adaptive mitigation is to design lightweight self-tuning priority-based mitigation strategies that autonomously determine in real time how to set priorities between interacting applications. This is not a trivial problem as it involves exploring trade-offs between the state to maintain to account for previous decisions and the accuracy of the inference method used.

C. Intent-based Management

Recent years have witnessed a growing interest in the community for intent-based networking[5]. The objective is to hide the complexity of the underlying managed infrastructure from the operators who can focus on expressing, in the form of high-level intents, what they want the network to do rather than how to do it. In particular, it alleviates the burden of slow and error prone manual configuration by enabling the development of adaptive management functionality that autonomously decides how to configure the network at run time based on some high-level operator objectives.

[5]https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-523_Intent_Definition_Principles.pdf

In the context of interaction management, a challenging issue is to develop a framework that enables the translation of these high-level intents into instructions that can be understood by the interaction management functionality. To achieve this objective, it is first essential to analyze what metrics to use to characterize the performance and effects of different mitigation actions. A possible approach is to focus on the properties presented in Section III-C in order to derive quantifiable metrics to assess the expected performance and to show how they can be linked to specific high-level objectives (*e.g.,* reduce downtime). We recently proposed an initial implementation of a northbound interface in [15]. Extending the interface with additional functionality to autonomously decompose relevant intents for the purpose of selecting mitigation strategies and tuning their parameters requires further work.

VI. Conclusions

To meet the requirements of new advanced networking approaches and services, rich management functionality is needed. This raises questions on how to facilitate the integration of a wide range of functions in an environment where potentially conflicting applications operate in parallel. The handling of application interactions is a complex issue that involves multiple challenging facets. Addressing these challenges is crucial before software-based networks can be deployed in the real. In this paper we review and analyze previous efforts in that direction and discuss key open issues for achieving automated and real-time management of interactions.

Acknowledgment

The work of Daphne Tuncer is supported by the Imperial College Research Fellowship Scheme. This research was also partly funded by the EPSRC KCN project (EP/L026120/1).

References

[1] M. Claeys *et al.*, "Hybrid Multi-tenant Cache Management for Virtualized ISP Networks," in *Journal of Network and Computer Applications (JNCA)*, vol. 68, pp. 28-41, June 2016.

[2] R. Landa, M. Charalambides, R.G. Clegg, D. Griffin, and M. Rio, "Self-Tuning Service Provisioning for Decentralized Cloud Applications," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 197-211, June 2016.

[3] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing Software Defined Networks," in *Proc. of NSDI'13*, vol. 13, pp. 1-13, 2013.

[4] X. Jin, J. Gossels, J. Rexford, and D. Walker, "CoVisor: A Compositional Hypervisor for Software-Defined Networks," in *Proc. of NSDI'15*, vol. 15, pp. 87-101, May 2015.

[5] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering in an ISP network," in *Proc. of SIGMETRICS '09*, New York, NY, USA, pp. 239-250, 2009.

[6] M. Charalambides et al., "Policy conflict analysis for diffserv quality of service management," in *IEEE Transactions on Network and Service Management*, vol. 6, no. 1, pp.15-30, March 2009.

[7] P. Sun, R. Mahajan, J. Rexford, L. Yuan, M. Zhang, and A. Arefin, "A network-state management service," in *Proc. of the 2014 ACM conference on SIGCOMM (SIGCOMM '14)*. pp. 563-574, 2014.

[8] A. Wang and J. Croft, "Automating SDN Composition: A Database Perspective," in *Proc. of the Symposium on SDN Research (SOSR'17)*, pp. 203-204, 2017.

[9] W. Wang, W. He and J. Su, "Redactor: Reconcile network control with declarative control programs In SDN," in *Proc. of the 24th IEEE International Conference on Network Protocols (ICNP'16)*, Singapore, pp. 1-10, 2016.

[10] A. AuYoung et al., "Democratic Resolution of Resource Conflicts Between SDN Control Programs," in *Proc. of the 10th ACM International on Conference on emerging Networking Experiments and Technologies (CoNEXT'14)*, pp. 391-402, 2014.

[11] R. Riggio, M. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rasheed, "Programming abstractions for software-defined wireless networks," in *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 146-162, June 2015.

[12] V. Heorhiadi, M. K. Reiter, and V. Sekar, "Simplifying software-defined network optimization using SOL," in *Proc. of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI'16)*, Santa Clara, CA, USA, pp. 223-237, 2016.

[13] S.H. Yeganeh, and Y. Ganjali, "Beehive: Simple Distributed Programming in Software-Defined Networks," in *Proc. of the Symposium on SDN Research (SOSR'16)*, pp. 1-12, 2016.

[14] G. Tesauro, "Reinforcement Learning in Autonomic Computing: A Manifesto and Case Studies," in *IEEE Internet Computing*, vol. 11, no. 1, pp. 22-30, Jan.-Feb. 2007.

[15] D. Tuncer, M. Charalambides, G. Tangari, G. Pavlou, "A Northbound Interface for Software-based Networks,," in *Proc. of the Internartional Conference on Network and Service Management (CNSM'18)*, Rome, Italy, November 2018.

Daphne Tuncer (e-mail: dtuncer@ic.ac.uk) is a Research Fellow in the Department of Computing at Imperial College London, UK. She received a Ph.D. from University College London (UK) in 2013 and a Diplome d'ingenieur de Telecom SudParis (France) in 2009. Her research interests are in the areas of software-defined and programmable networks, adaptive network resource management and monitoring, and multimedia content distribution.

Marinos Charalambides (e-mail: marinos.charalambides@ucl.ac.uk) is a senior researcher at University College London. He received a BEng in Electronic and Electrical Engineering, a MSc in Communications Networks and Software, and a Ph.D. in Policy-based Network Management, all from the University of Surrey, UK, in 2001, 2002 and 2009, respectively. His research interests include network programmability, adaptive resource management, content delivery and network monitoring.

George Pavlou (e-mail: g.pavlou.ucl.ac.uk) is Professor of Communication Networks in the Department of Electronic and Electrical Engineering, University College London, UK. He received a PhD in Computer Science from University College London, UK. His research interests focus on networking and network management, including aspects such as autonomic networking and software defined networks. He is the chief editor of the bi-annual IEEE Communications network and service management series and in 2011 he received the Daniel Stokesbury award.