A Flexible Multi-Cloud Provider Monitoring Solution

Journal:	*IT Professional*
Manuscript ID	ITPro-2020-05-0049
Manuscript Type:	General Interest
Date Submitted by the Author:	01-May-2020
Complete List of Authors:	Sabbioni, Andrea; Università di Bologna, Department of Computer Science and Engineering Bujari, Armir; Università di Bologna, Computer Science and Engineering Foschini, Luca; Universita degli Studi di Bologna, Dipartimento di Informatica - Scienza e Ingegneria (DISI) Corradi, Antonio; University of Bologna, DEIS
Keywords:	PaaS, middleware, Multi-cloud, Monitoring

A Flexible Multi-Cloud Provider Monitoring Solution

Andrea Sabbioni, Armir Bujari, *Member, IEEE,* Luca Foschini, *Member, IEEE,* and Antonio Corradi, *Senior Member, IEEE,*

***Abstract*—Cloud computing has had a profound impact on IT professionals and businesses, offloading the overhead of configuration and management of resources to the cloud provider. While a lot of research has been conducted to optimize and propose (new) in-cloud service delivery models, yet, service outages are the norm rather than an exception. In this context, the multi-cloud integration and deployment pattern is an appealing preposition, especially for applications that must meet high availability requirements. A distributed cloud deployment model coupled with a monitoring solution spanning multiple domains could help in a timely identification and isolation of faulty components, alleviating service discontinuity problems. Pursuing this goal, we propose an extension to NoMISHAP, a Platform as a Service (PaaS) multi-cloud middleware, introducing a component-based monitoring solution for use in distributed cloud infrastructures. Collected results show the effectiveness of our proposal and its ease of adoption for IT management tasks.**

***Index Terms*—PaaS, middleware, Multi-cloud, Monitoring.**

I. Introduction

Nowadays, the efficient deployment, and maintenance of business services has gained a strategic importance for both IT professionals and businesses. In this context, the Platform as a Service (PaaS) paradigm and the emerging technological ecosystem promises to reduce and to facilitate the evolution of business services. This cloud computing model provides the user with a development framework, while at the same tine alleviates the burden of dealing with management and control of the underlying infrastructure [1].

Over the years cloud providers have developed advanced and heterogeneous PaaS services, allowing a differentiation between the different vendors, potentially causing a lock-in effect for the customer. At the same time, the availability of the workloads hosted on cloud platforms is becoming more and more relevant for many businesses, and any shortage or service discontinuity can cause substantial losses. To address the above challenges, both academia and industry have been largely focused on solutions ranging from novel architectures and deployment models, programming abstractions, to specific tools aiding service lifecycle management. In particular, academic research has been focused more on theoretical grounds, investigating areas such as service portability and composition, efficient service placement and migration strategies, and software engineering aspects [2], [3]. On the industry side, instead, cloud providers have put in place best practices for the implementation of services; however, all these solutions typically are limited to a single cloud providers and are not suitable to address large-scale and multi-region deployment scenarios [4]. Despite the cloud services market's steady growth, no cloud solution provider has really yet proven to be immune from outages and major episodes of service disruptions are the norm rather than an exception [5].

A. Sabbioni, A. Bujari, L. Foschini and A. Corradi are with the University of Bologna, Computer Science and Engineering Department, Bologna, Italy

The relevant point of view for cloud-based application developers, instead, should be a more external perspective where, to overcome such service outages, they could leverage and use together multiple services provisioned in multiple cloud environments. The outcome is a more resilient and lock-in free multi-cloud ecosystem. Indeed, a promising path that is being considered is the use of the so-called multi-cloud integration pattern [6], a solution that simultaneously exploits multiple cloud platforms for service delivery. While this approach usually implies an overhead in terms of orchestration complexity, the implementation of a practical multi-cloud approach could facilitate the reach of important business objectives like high availability, failover and lock-in avoidance. In this context, an effective and complete monitoring solution spanning different layers of a service plays a key role in fulfilling the users requirements, serving as the basis for an (semi)automatic identification and mitigation of faults [7].

To this end, we set to design and implement a pervasive monitoring solution for use in the multi-cloud PaaS context. While preserving the general aspect of our study and without loss of generality, we chose to extend NoMISHAP [8], a state-of-the-art multi-cloud middleware solution, by making it more robust to faults and able to sustain possible high load peaks. The proposed solution aims to provide a complete monitoring suite able to collect information across all the layers of a multi-cloud PaaS. The gathered monitoring data can serve as input to an (semi)automatic failure recovery procedure and/or cloud operators, aiding to recover and/or prevent potential hazardous conditions in an efficient way. Finally, it is noteworthy to point out, that our proposal does not substitute existing ones offered by cloud providers. On the contrary, it integrates them and fills-in eventual gaps, providing additional information spanning multiple layers.

II. Sidebar: Toward Multi-Cloud PaaS Support

Most cloud offerings are equipped with complex monitoring solutions, but all these tools are usually tailored and locked-in to the provider infrastructure and cannot be ported and/or easily tapped to span heterogeneous cloud environments [2]. Tackling the issue, several proposals pursuing a multi-cloud

approach can be found in literature. A seminal effort in this direction is presented by the open source project OpenNebula [9]. The solution offers a cloud computing toolkit for managing heterogeneous distributed datacenter infrastructures. It can orchestrate storage, network, monitoring, and security technologies to deploy multi-tier services as virtual machines on distributed infrastructures, combining both datacenter and remote cloud resources. This project presents a lot of interesting features and a powerful toolkit for multi-cloud environments. However, it targets integration at the IaaS layer only.

Pursuing a similar objective, different academic projects have addressed the challenging task of provisioning software stacks, libraries for frameworks aimed for the deployment, monitoring and adaptation of cloud-based systems at the IaaS and/or PaaS layers. FraSCAti is a solution which relies on the extended service component architecture, a technology agnostic standard for developing and deploying distributed service-oriented applications, to deploy federated multi-cloud PaaS infrastructures [10]. Its open service model allows FraSCAti to leverage on both PaaS infrastructure and the SaaS applications hosted on top of it.

soCloud extends the FraSCAti execution engine inheriting its capabilities of extending across multi-cloud PaaS environments. In addition, it introduces new features facilitating portability, provisioning, resilience and high availability of services across multiple clouds [11].

Cloud4SOA introduces a broker-based architecture, enabling a scalable approach to heterogeneous PaaS offerings integration in terms of semantic interconnection between different providers sharing the same technology [12]. The architecture is equipped with management and monitoring capabilities providing the appropriate flexibility to handle either public or private deployment models.

Both FraSCAti and Cloud4SOA are elaborated examples proposing a multi-layer integration approach for the multi-cloud domain, however, they require the deployment of additional physical resources, either on-site or off-site, hosting the control logic.

Addressing heterogeneous multi-cloud environments, mOSAIC focuses on both IaaS and PaaS layers by allowing applications to specify their service requirements through an ontology [13]. The proposal relies on a brokering mechanism exploited to search for the best set of services meeting application requirements. The main outcome of the mOSAIC project is a common communication API for multi-cloud resources.

NoMISHAP is a middleware solution aimed at providing a transparent support for high availability, exploiting multiple cloud PaaS providers simultaneously [8]. The proposal achieves its promise by introducing an abstraction and adaption layer used to simplify and unify the access to services available on the underlying PaaS. This feature allows cloud developers to concentrate only on essential core business code, developed once and deployed on multiple PaaS solutions. Compared to the prior discussed proposals, NoMISHAP poses a lower barrier of entry, allowing for the integration of individual PaaS services spanning multiple environments through the use of lightweight proxies. The proxy could be provisioned *in situ* or hosted elsewhere as a third party service.

III. The NoMISHAP Monitoring Proposal

Herein we discuss the extensions made to NoMISHAP. To this end, we start by identifying the monitoring data sources and than discuss the approach and tools adopted to extract them. We anticipate our integration approach makes use of an adaptation layer and associated components allowing us to integrate monitoring services/components, providing a uniform and logically centralized data presentation layer.

A. Data Sources

To maintain a complete and consistent view of the PaaS layer, we need to collect metrics pertaining to the service components the application relies on and the NoMISHAP middleware components deployed on the PaaS. Unfortunately, not all PaaS providers expose state statistics related to the services. Filling in this gap, we can consider as a good representation of performance and state, the historical data transparently collected by the proxy itself. The monitoring data are then forwarded to a logically centralized data aggregation service.

Another source of data are the performance metrics as perceived by the client proxy while forwarding user requests to the appropriate PaaS. This intermediary component collects and computes metrics related to every call executed over the NoMISHAP middleware against the PaaS proxies. The gathered monitoring data are then periodically sent for digestion to a logically centralized aggregation service (later on). Finally, another piece of the puzzle in the data gathering effort is on the client side. Collecting metrics on the client side enables us to estimate the perceived end-to-end QoS and can help detect incidents and/or faults in part(s) of our infrastructure. These end-to-end measurements can in principle be compared with the prior ones, denoting different logical segments of the communication, and could help to diagnose and pinpoint performance issues.

While data collection related to the communication path is of paramount importance, one should not neglect health data related to the actual functional components of the architecture. To this end, depending on the monitoring features provisioned by the PaaS environment, different solutions are viable for extracting health information concerning the proxy components itself. In scenarios where the PaaS solution exposes status data related to the services, our monitoring solution can tap to the monitoring service available at the PaaS layer. Otherwise, the proxy can itself collect metrics about state and performance while handling requests such as measuring their response times.

B. A Component-based Approach

Figure 1 shows the integration approach and components used to collected data from the data sources, feeding them to an analytics components. Starting from the bottom layer, to collect the data on the PaaS side, we rely on an adapter software component which interfaces with the PaaS service layer. The component is capable of acquiring the metrics from the provider monitoring service(s) - currently extended

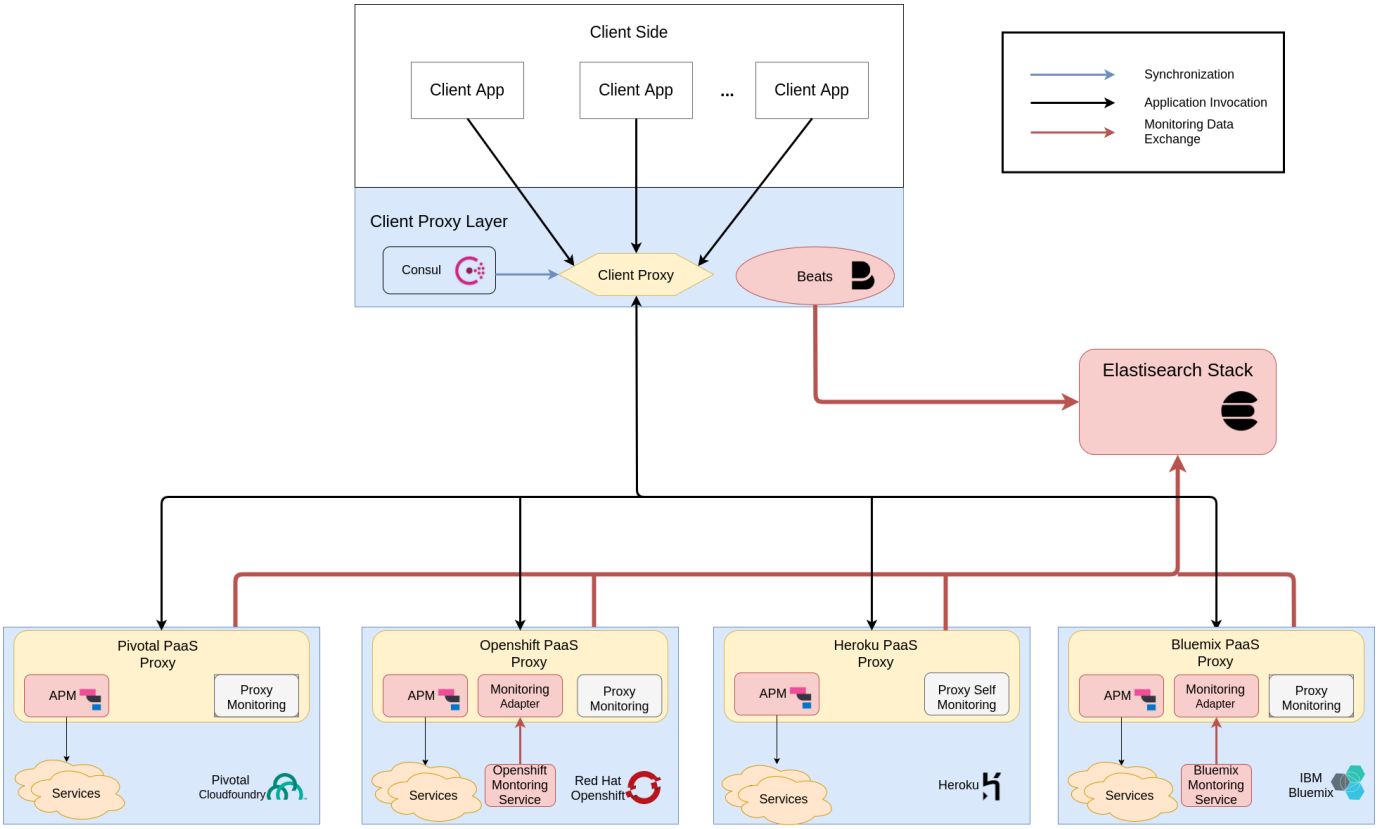

Fig. 1: The proposed monitoring solution. From the bottom up, one finds the PaaS proxy component tailored to the specific PaaS environment. The component is extended to contemplate for monitoring functionalities pertaining to the PaaS component(s) used by the application and the proxy itself (gray). Moving up is the client-proxy component, mediating user requests toward the NoMISHAP middleware. The component contemplates a monitoring module used to gather usage and component health data. Aside, is the logically centralized data aggregation service consisting of a (distributed) Elasticsearch service.

to support some major providers such as IBM, Pivotal [14], Openshift [15], and Heroku [16] - successively forwarding them to a data aggregation service. The data at this stage, if desirable, might be subject to some preliminary computation and/or filtering.

In order to collect data pertaining to the proxy itself, the component was extended to contemplate for a data gathering module. This additional module, evidenced in Fig. 1, could also be distributed elsewhere, acting as a standalone component. To this aim, we exploit the Elasticsearch APM module which has the capability of intercepting REST calls passing through the proxy and executed against PaaS services, producing advanced statistics like errors and response times. All the data gathered are successively sent to an APM server which is responsible for data aggregation and successively feed them to a logically centralized point of aggregation.

The available PaaS environments comprising the multi-cloud solution, notify their presence and register with the client proxy, acting as an intermediary dispatching user requests. This layer needs an up-to-date view on the underlying resources, available PaaS environments. To this end, the client-proxy relies on a synchronization service embodied by *Consul.io*, a service networking solution used to connect and secure services across any runtime platform. This proxy acts as an entry point for nearly all operations, and its operational performance is critical for the overall throughput and health of the multi-cloud environment.

To fetch advanced metrics on nodes hosting the synchronization service, we exploit the Beats component present in Elasticsearch. Beat services, are additional modules that can be installed on nodes, allowing to intercept and expose advanced and complete metrics about the state of a hosting node and eventual services installed. The extracted metrics are then reliably sent to the data aggregation service which can be consulted on demand.

The proposal is not complete without provisioning an analytics engine capable of visualizing and reacting to changes in the environment. Also, it is desirable that this engine itself could scale on a per-need basis. To this aim, this functionality resides and is embedded on the Elasticsearch analytics. In our proposal, Elasticsearch constitutes the final endpoint responsible for the gathering, computation and storage of the metrics of interest, enabling the visualization, reporting and advanced analysis in a (near) real time fashion. Thanks to advanced sharding techniques, Elasticsearch is able to scale, through a policy-based engine, and replicate functional components across compute clusters. At the end of the provisioned pipeline stands Kibana, a data visualization tool equipped with

	#Nodes	Provider	Typology	RAM/Host (GB)	#CPU/Host
Locust	3	Garr Cloud	Virtual Machine	4	2
Consul	5	2 Azure Public cloud 3 Private Hosting	Virtual Machine	8	2
Elastic Stack	2	Private Hosting	Docker Container	16	4
Pivotal	2	Pivotal Web Services	Cloud Foundry Container	0.5	1
OpenFaaS on OpenShift	2	Private Hosting	Kubernetes Container	1	2
Bluemix	2	IBM Bluemix	Cloud Foundry Container	0.25	1
Heroku	1	Heroku	Dynos	0.5	1

TABLE I: Characteristics of each cloud provider solution.

advanced graphics and featured maps. All these services, part of the Elasticsearch stack, are capable also of monitoring the hosting infrastructure, providing advanced information and alerts on the cluster(s) status.

As a final note, when dealing with a potentially large amount of data, direct forwarding of the information to the Elasticsearch cluster might not be feasible. To address this potential issue, one could rely on the Elasticsearch Logstash service, subjecting the data sources to a (pre)processing pipeline by applying transformations, contributing to the scalability of the whole monitoring infrastructure during peaks.

IV. Experimental Assessment

In the following, we discuss the multi-cloud testbed and its configuration used to asses our proposal. We then conclude by presenting some experimental results focusing on the core components. The NoMISHAP source code and the testbed configuration used for the assessment can be found in [17].

A. Testbed and Configuration

To assess the capabilities of our proposal, we set up a test infrastructure emulating a real scenario where a number of concurrent users issue requests against a PaaS services mediated by the NoMISHAP middleware. For the purpose of this experimentation, we consider a PDF conversion service provisioned in each of the considered cloud environments. Concerning the multi-cloud PaaS environment, we chose to rely on three top solutions available in the market shown in Fig. 1. The characteristics of the respective testbeds are summarized in Table I.

To simulate a group of concurrent users on the distributed platform, we exploit Locust, a distributed tool used to execute infrastructure load test. To this end, we set up three Locust nodes and locally installed the client proxy which transparently mediates user requests issued against the NoMISHAP middleware.

The test lasted 12 hours reaching a peak of 5000 concurrent requests/s. To evidence the benefits of our proposal, we simulate a fault-like behaviour in one subsystem, hosting the client-proxy at 16:10. This fault corresponds to a sudden drop on the access link capacity going from 100 Mbps to 200 Kbps. Through this, we would like to asses our proposals both in terms of being capable of identifying the change in behaviour and in pinpointing the source of anomaly. We expect that the metrics gathered from all the subsystems coupled with the knowledge of the relationships between the components of the infrastructure could enable a fast discovery of the cause.

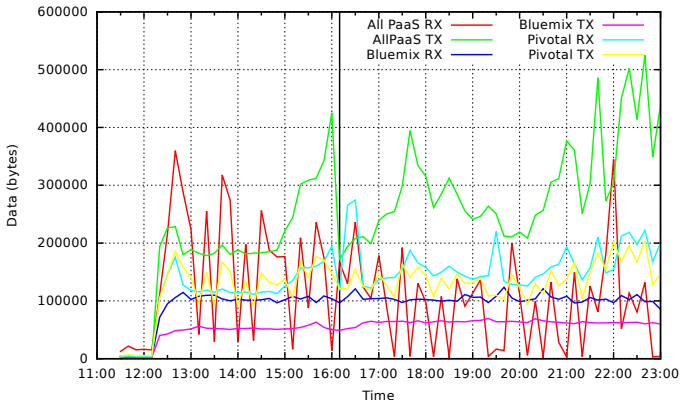

Fig. 2: Average response time of the PaaS services. The vertical line at 16:10 denotes the start time of the capacity drop phenomena.

B. Results

Let us now delve into the behaviour of the monitoring system and its thorough assessment to verify that it fulfills our design objective.

The first component is the PaaS services layer provided through the different environments. This layer can contribute to an increase in response times when one or more services go down but also depending on the service resource quota allocation and sustained load.

Thanks to the monitoring stack added to the PaaS proxy, we are able to monitor even the services which are not under our direct control. Figure 2 shows the individual service load evolution for the duration of the experiment. The client proxy forwards user requests in a round robin fashion to the available PaaS platforms comprised in the experimental multi-cloud platform. Overall, the trend shows periodic and justifiable behaviour of services inside the single PaaS with periodic fluctuations in terms of received and transmitted data.

Going upward in the invocation chain, we find the PaaS proxy component. Figure 3 shows, among other data, the (ingress) average traffic load the component is subject to, with an evident decrease in volume at 16:10. The cause for this reduction in traffic is to be found upward in the chain call, which considering the experimental setup, points to the client network. We report that despite the operational constraints imposed in the client network, the client proxy synchronization mechanism consumes a small amount of bandwidth and the component is capable of guaranteeing operational continuity.

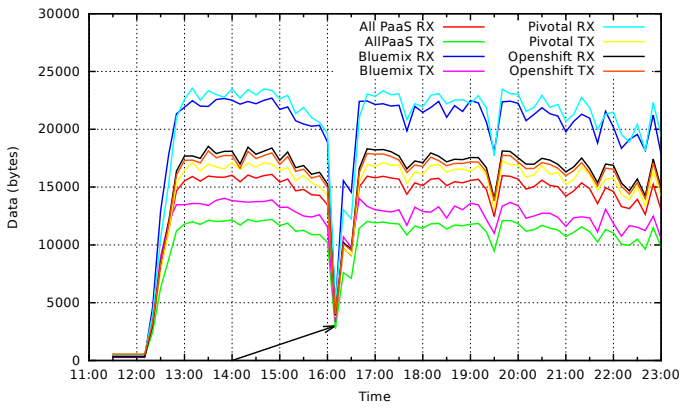

Fig. 3: Average network traffic registered by PaaS proxies.

V. Conclusion

In this article, we presented and discussed a monitoring solution for use in distributed, multi-cloud environments. To asses and validate the design rationale, we presented an experimental analysis involving a realistic testbed consisting of three heterogeneous PaaS environments. The proposal can be further extended to provide for intelligent, rule-based mechanisms capable of handling (semi)automatic fault mitigation/migration. Yet another interesting on-going development is that of extending the architectural principles to those of edge computing. In this case, one additional and challenging research question is that of extending the solution to contemplate for a Function as a Service layer.

References

[1] Mell, P. and Grance, T. Evaluation of Cloud Computing Services Based on NIST SP 800-145, NIST Spec. Publ. 800 7, 2011.
[2] Aslam, S. *et al.*. Information collection centric techniques for cloud resource management: Taxonomy, analysis and challenges. Journal of Network and Computer Applications, vol. 100, pp. 80–94, 2017.
[3] Jabbarifar, M., Shameli-Sendi, A and Kemme, B. A scalable network-aware framework for cloud monitoring orchestration. Journal of Network and Computer Applications, vol. 133, pp. 1–14, 2019.
[4] Amazon AWS [Online]. Patterns for High Availability. Availalbe: http://en.clouddesignpattern.org/index.php/Main_Page#Patterns_for_High_Availability
[5] Data Economy [Online]. Outages. Downtime. System Failures. 2019's IT Meltdowns. Available: https://data-economy.com/outages-downtime-system-failures-2019s-it-meltdowns/
[6] Petcu, D. Multi-Cloud: Expectations and Current Approaches. In Proc. of the International Workshop on Multi-cloud Applications and Federated Clouds, New York, NY, USA, 2013.
[7] Ghazi *et al*. Cloud monitoring: A review, taxonomy, and open research issues. Journal of Network and Computer Applications, vol. 98, 11-26, 2017.
[8] Acquaviva, L. *et al.* NoMISHAP: A Novel Middleware Support for High Availability in Multicloud PaaS. IEEE Cloud Computing, vol. 4, no. 4, 2017.
[9] Milojičić, D., Llorente I. M. and Montero, R. S. "OpenNebula: A Cloud Management Tool," in IEEE Internet Computing, vol. 15, no. 2, pp. 11-14, 2011.
[10] Paraiso, F., Haderer, N., Merle, P., Rouvoy, R. and Seinturier, L. A Federated Multi-cloud PaaS Infrastructure. In Proc. of the IEEE International Conference on Cloud Computing, pp. 392–399, 2012.
[11] Paraiso Fawaz and Merle, P. soCloud: a service-oriented component-based PaaS for managing portability, provisioning, elasticity, and high availability across multiple clouds. Computing, vol. 98, no. 5, pp. 539–565, 2016.
[12] F. Dandria, S. B. Cloud4SOA: Multi-cloud Application Management Across PaaS Offerings. In Proc. of the International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, pp. 407-414, 2012.
[13] Zeginis *et al*. A user-centric multi-PaaS application management solution for hybrid multi-Cloud scenarios. Scalable Computing: Practice and Experience, vol. 14, 2013.
[14] Cloud Fundry [Online]. Available: https://www.cloudfoundry.org/
[15] OpenShift [Online]. Available: https://www.openshift.com/
[16] Heroku Cloud Application Platform [Online]. Available: https://www.heroku.com/
[17] NoMISHAP Repository [Online]. Available: https://github.com/NoMishap.