Trustworthy and Resilient Monitoring System for Cloud Infrastructures

Smruti Padhy, Diego Kreutz, António Casimiro, Marcelo Pasin
LASIGE, Faculty of Sciences, University of Lisbon, Portugal
{spadhy, kreutz}@lasige.di.fc.ul.pt, {casim, pasin}@di.fc.ul.pt

ABSTRACT

Current monitoring systems for cloud infrastructures are based on local, centralized or hierarchical model approaches such as HP Openview and ArcSight. Additionally, they do not look deep into resilience and delivering trustworthy data of its own services under crash or Byzantine failures caused by attackers or any other kind of sources. This work proposes a fault and intrusion tolerant monitoring system for cloud computing infrastructures. We assume a Byzantine failure model and use state machine replication for providing the trustworthy and resilient monitoring service.

Categories and Subject Descriptors

C.2.3 [**Network operations**]; C.2.4 [**Distributed Systems**]; C.4 [**Fault-tolerance**]

General Terms

Design, Management, Reliability, Security

Keywords

Byzantine faults, Monitoring, Publish-Subscribe, Cloud infrastructure

1. MOTIVATION

Cloud computing is becoming popular for vast available computing resources on clouds at affordable price and hassle free installation. A typical data center deploys several machines to provide IaaS, SaaS and PaaS. Besides, it has control, monitor and management services to its internal network such as LDAP, DNS, SNMP, Puppet and CFEngine. These services are responsible for the correct operation of cloud infrastructure and its security. It is envisioned that cloud services are going to be used in public critical infrastructures such as grid, healthcare and other strategic applications. Concerns about quality of service and quality of protection when using IaaS for these environments become very critical because companies and people rely their services on these cloud infrastructures expecting 100% of uptime. Thus, they need to be monitored in a resilient and trustworthy way for security and availability.

A typical cloud infrastructure monitoring system comprises probes or agents connected to a console through a centralized system for event message gathering. One existing solution is ArcSight Enterprise Security Manager. It works in a centralized way collecting information, basically logs, from various devices and applications, and generating alerts on the occurrence of some critical events. HP Openview and Microsoft SCOM are examples of similar tools. Further, there are also some more specific IaaS monitoring tools such as Cloudkick, Zenoss, Amazon Cloudwatch, LogicMonitor and CloudSec. These monitoring tools also work in centralized models and do not ensure availability of their services, being a single point of failure or attack. To address this problem, as shown in Fig. 1, we propose to design a Fault and Intrusion Tolerant (FIT) monitor with replication to ensure the trustworthiness and resiliency of the monitoring system itself. We use a topic-based event-based Publish-Subscribe (P-S) system for event message dissemination. Probes become publishers while consoles act as subscribers. The FIT monitor works as a reliable framework for delivering data from publishers to subscribers. To ensure Byzantine fault tolerance in our proposal we use state machine replication. This makes the monitoring system resilient to any kind of faults, allowing system and network monitoring to be trustworthy.

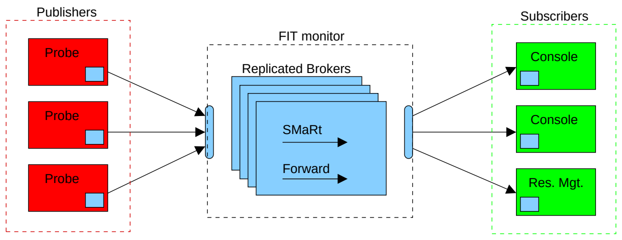

Figure 1: Abstraction of FIT monitoring system

2. SYSTEM MODEL

Probes (or agents), FIT monitor, and consoles are major components of the proposed architecture for monitoring IaaS. Further, we make the following assumptions for our system: **Network**. All probes and consoles can reach each FIT monitor replica. **Synchrony**. Timed-asynchronous system model where messages can be arbitrarily delayed and processes have access to local clocks. **Faults.** Byzantine failure/fault model where any of the FIT monitor replicas may crash or behave arbitrarily. The monitoring system can tolerate upto f faulty replicas. There can be event message delay or loss due to link failure or some malicious attacker. To ensure integrity of the event message generated by a probe, we assume that they are able to digitally sign messages with its private key (e.g. RSA).

3. FIT MONITORING SYSTEM

We explain the building blocks of the proposed FIT monitoring system and detail also its architecture. Fig. 1 represents its abstract architecture while Fig. 2 shows the internal composition of a broker.

3.1 Building Blocks

Event handling: We propose to use the P-S paradigm for communication and event handling [3]. Probes, event brokers and consoles can be mapped to a P-S system. A probe publishes messages to event brokers. The FIT monitor forwards event messages to its subscribers in a trustworthy way. Consoles, resource managers and different applications subscribe themselves on the brokers to receive specific information. In this work, we focus on topic-based P-S. There are several works on search methods; topic-based, content-based and type-based, in P-S systems. However, only a few works addressed the fault-tolerance in P-S system. In [4], the authors provide a P-S algorithm that tolerates up to δ crash failures. However, as far as we know, BFT event brokers have not been addressed in the literature. Our broker design for BFT will contribute to the P-S system research.
Client side: A probe is a program that generates a message on the occurrence of events such as security threats and system state changes like network flow statistics, storage space usage and processor usage. A console is a software or hardware component that receives monitoring messages for displaying or alerting system administrators. A subscriber may also be a resource manager or an event correlation engine. A console is also integrated with a voter interface. It performs a voting on messages received from different brokers, for the same event, before delivering the result to users. Both probes and consoles are broker's clients. Probes generate event messages which are pushed into the event broker through a previously defined event channel. Each console subscribes to one or more specific channels of the FIT monitor to receive the messages of its interest.
Server side (FIT monitor): An event broker has as primary job to manage all received messages from different probes and subsequently propagate them to subscribers in a trustworthy way.
Intrusion tolerance: FIT monitor acts as a broker to all event messages send by probes. We replicate the broker to provide a trustworthy and fault tolerant monitoring service. To incorporate Byzantine fault tolerance (BFT) into the FIT monitor, we propose to use state machine replication which allows replicas to perform any operation, provided they are deterministic. There are well known BFT algorithms as well as challenges that still need to be addressed [2].

3.2 Broker internals

Fig. 2 shows the internal architecture of a broker. Its components are: interface, forwarding protocol, ordering protocol and topic-based filter. The interface provides basic methods to create an event channel between publisher and broker. Consoles may subscribe themselves into each existing channel. Based on QoS requirements of event messages, the interface allows a user to specify the channel properties such as order, urgency and fault-tolerance.
As can be seen, the broker supports Byzantine tolerant (path I) as well as crash tolerant (path II) event channels. In the case of crash-tolerant channels, all messages are just forwarded. For the Byzantine case, we use SMaRt [1], which implements a BFT protocol.

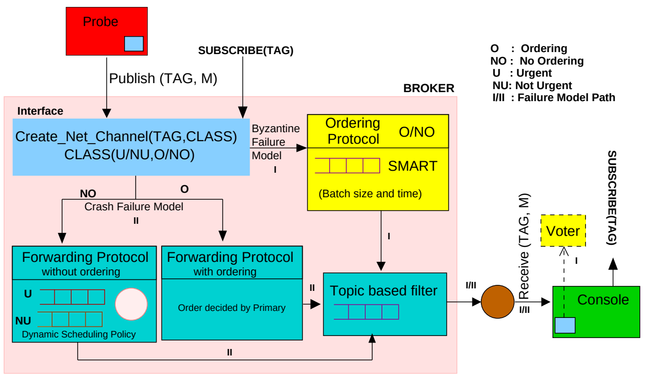

Figure 2: Broker internals

4. DISCUSSION

We plan to validate the proposed solution in our cloud infrastructure. We will use probes for monitoring IaaS, such as OpenStack and OpenNebula, and consoles to receive and display event messages sent by probes. Crash failures and attacks will be injected into the FIT monitoring system, in order to evaluate its effective resilience and trustworthiness. The results will be of particular interest to Telecoms which have a special interest in improving the reliability of their monitoring systems.

Acknowledgement

This work is supported by the EC, through SECFUNET project IST-FP7-STREP-288349 and by the FCT, through TRONE project CMU-PT/RNQ/0015/2009 and the Multi-annual and CMU-Portugal programmes.

5. REFERENCES

[1] A. Bessani, et. al. BFT-SMaRt - High-Performance Byzantine-Fault-Tolerant State Machine Replication, 2011. `http://code.google.com/p/bft-smart/`.

[2] A. Bessani. From Byzantine Fault Tolerance to Intrusion Tolerance (A Position Paper). *5th Workshop on Recent Advances in Intrusion-Tolerant Systems (WRAITS), DSN*, 2011.

[3] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Survey*, 35:114–131, June 2003.

[4] R. S. Kazemzadeh and H.-A. Jacobsen. Reliable and Highly Available Distributed Publish/Subscribe Service. In *Proceedings of 28th IEEE International Symposium on Reliable Distributed Systems*, pages 41–50, 2009.