BANCA D'ITALIA

EUROSISTEMA

# Mercati, infrastrutture, sistemi di pagamento

(Markets, Infrastructures, Payment Systems)

## TIPS: a zero-downtime platform powered by automation

by Gianluca Caricato, Marco Capotosto, Silvio Orsini and Pietro Tiberi

October 2022

Number 28

BANCA D'ITALIA
EUROSISTEMA

# Mercati, infrastrutture, sistemi di pagamento
(Markets, Infrastructures, Payment Systems)

## Approfondimenti
(Research Papers)

# TIPS: a zero-downtime platform powered by automation

by Gianluca Caricato, Marco Capotosto, Silvio Orsini and Pietro Tiberi

Number 28 – October 2022

*The papers published in the 'Markets, Infrastructures, Payment Systems' series provide information and analysis on aspects regarding the institutional duties of the Bank of Italy in relation to the monitoring of financial markets and payment systems and the development and management of the corresponding infrastructures in order to foster a better understanding of these issues and stimulate discussion among institutions, economic actors and citizens.*

*The views expressed in the papers are those of the authors and do not necessarily reflect those of the Bank of Italy.*

*The series is available online at www.bancaditalia.it.*

*Printed copies can be requested from the Paolo Baffi Library: richieste.pubblicazioni@bancaditalia.it.*



ISSN 2724-6418 (online)
ISSN 2724-640X (print)

Banca d'Italia
Via Nazionale, 91 - 00184 Rome - Italy
+39 06 47921

*Designed and printing by the Printing and Publishing Division of the Bank of Italy*

# TIPS: A ZERO-DOWNTIME PLATFORM POWERED BY AUTOMATION

by Gianluca Caricato,* Marco Capotosto,* Silvio Orsini* and Pietro Tiberi*

**Abstract**

Modern ICT services are characterized by pervasive connectivity and the need for applications that are always on (24/7); a new way of designing and operating such services is therefore essential. To cope with these requirements, a proper system architecture design and an effective and efficient test phase are fundamental.

This work describes the zero-downtime architecture underlying the TIPS (Target Instant Payment Settlement) design, which makes it possible to achieve the expected levels of availability and reliability. This architecture enables standard maintenance activities to be carried out on TIPS during business hours without any service interruptions. However, some heavy maintenance activities still require a planned downtime; in order to eliminate this downtime, a project to adopt a three-site architecture with an active-active-active configuration is underway. This footprint will give TIPS levels of availability of the order of '5 nines' (i.e., 99.999% of the time, corresponding to 5 minutes' outage per year).

Furthermore, the increased complexity of the system and the growth in the number and sophistication of threats have led to intensive use of automation throughout the entire TIPS service lifecycle. Automation allows the quality of the TIPS software to be less defective by catching bugs at early stage, thereby reducing human errors and avoiding configuration drifts.

Following an initial discussion about the architectural choices and the corresponding levels of availability, this paper describes the use of automation in the specific area of non-functional tests (NFTs), which guarantee robustness, business continuity and security.

**Keywords**: TIPS, Availability, Automation.

---

* Bank of Italy, IT Operations Directorate.

## Sintesi

I moderni servizi ICT sono caratterizzati da connettività pervasiva e dalla necessità di applicazioni sempre attive (H24/7D); un nuovo modo di progettare e gestire tali servizi è obbligatorio. Per far fronte a queste esigenze diventa fondamentale una corretta progettazione dell'architettura di sistema e una efficace ed efficiente fase di test.

Questo lavoro descrive l'architettura *zero-downtime* alla base del design di TIPS (*Target Instant Payment Settlement*), che rende possibile raggiungere i livelli attesi di disponibilità ed affidabilità. Tale architettura consente l'esecuzione di attività di manutenzione standard su TIPS durante le ore lavorative senza interruzioni del servizio. Tuttavia, alcune attività di manutenzione pesante richiedono ancora un fermo di sistema pianificato; al fine di eliminare questo *downtime*, è in corso un progetto per l'adozione di un'architettura a tre siti con configurazione attivo-attivo-attivo. Ciò consentirà di raggiungere una disponibilità dell'ordine dei "5 nove" (ovvero 99,999% del tempo, corrispondente a 5 minuti di interruzione all'anno) per il servizio TIPS.

Inoltre, la maggiore complessità del sistema e la crescita del numero e della sofisticatezza delle minacce, hanno portato a un uso intensivo dell'automazione durante l'intero ciclo di vita del servizio TIPS. L'automazione consente di migliorare la qualità del software TIPS rilevando eventuali bug sin dalle prime fasi di sviluppo, riducendo gli errori umani ed evitando errori di configurazione.

In questo articolo, dopo una disamina delle scelte architetturali e dei livelli di disponibilità che ne conseguono, viene descritto l'utilizzo dell'automazione nell'area specifica dei test non funzionali (NFT), che garantiscono la robustezza, la continuità operativa e la sicurezza del servizio.

# CONTENTS

# 1 Introduction

Target Instant Payment Settlement (TIPS) (Renzetti *et al.* 2021), (Arcese, D. Di Giulio and V. Lasorella 2018) is a platform providing an 24H/7D active ICT service, designed with tight constraints in terms of non functional requirements like continuous availability,[1] recoverability[2] and resiliency.[3] While the traditional ICT organization is based on shifts, strict service time and maintenance windows, performing maintenance activities while the service is active with *zero downtime* requires a new approach.

Operational continuity with zero downtime can be achieved by combining an appropriately redundant architecture - capable of overcoming the loss of some components or a whole processing site with no service disruption - with an unmanned approach to the operational management of the ICT activities making extensive use of automation. In fact, in a zero-downtime environment, the use of personnel must be limited as much as possible to keep costs under control and reduce operational risks.

This paper is composed of two main parts, describing respectively: (i) the service continuity model implemented with the *zero-donwtime* architecture currently used in TIPS, along with its expected evolution; (ii) the automation processes, with a focus on the execution of non functional tests, particularly crucial in ensuring survivability[4] and security.

# 2 Zero-downtime architecture

High availability is when a system has an uptime which is significantly higher than the amount of downtime and so the service can promise a high measure of availability. In practice, high availability can be achieved by removing any single points of failure and ensuring redundancy within the system. When a planned or unplanned outage does occur, the system has to react and redirect the workload to another component which can accomplish the same task. When the adverse event corresponds to the loss of a whole processing site, the amount of time necessary to reactivate the service is controlled by a project parameter known as Recovery Time Objectve (RTO). The RTO is defined by ISO standard as the targeted duration of time and a service level within which a business process must be restored after a disaster (or disruption) in order to avoid unacceptable consequences associated with a break in business continuity (“ISO 22301:2019. Business continuity standard. Implementation guide” 2019).

Traditionally, technical architectures aimed at ensuring business continuity have been based on the so called Active-Passive (AP) model. In this case, the primary data center is active while the secondary one is on stand by. The latter data center can be brought up within a reasonable time if the primary data center fails. As the expression “active-passive” suggests, at a given point in time

1. The amount of time a service is actually operating expressed as the percentage of total time it should be operating according to the SLA. High-availability systems (24*7) may report availability in terms of minutes or hours of downtime per year. Availability is typically measured as a percentage of the time a system is expected to be available (e.g. 99.999 percent: "five nines"). The amount of time the service is available is often referred to as uptime, otherwise downtime.

2. The ability to overcome a momentary failure in such a way that there is no or negligible impact on end-users. It could be as simple as an automatic logical recovery from a physical device to a spare one (Redundant Array of Independent Disk – RAID), and as complex as having an entire data center switch over with no loss of data or transactions. Recoverability also includes protocols to retry attempted reads and writes out to disk or tape, as well as retry of transmissions down network lines.

3. The ability to prepare for and adapt to changing conditions and withstand and recover rapidly and automatically from disruption. Resilience includes the ability to withstand and recover from deliberate attacks, accidents, or naturally occurring threats or incidents.

4. The ability of a system to provide essential services in the presence of attacks and failures, and recover full services in a timely manner (“Survivable Network Systems: An Emerging Discipline” 1997). A more detailed definition is in NIST SP 800-160 Vol.2 Rev. 1: the ability of a system to minimize the impact of a finite-duration disturbance on value delivery (i.e. stakeholder benefit at cost), achieved through the reduction of the likelihood or magnitude of a disturbance; the satisfaction of a minimally acceptable level of value delivery during and after a disturbance; and/or a timely recovery. (“NIST SP 800-160 Vol. 2 Rev. 1”, n.d.)

only one of the two data centers is active. This configuration requires human intervention in two crucial phases: the decision whether to reactivate the systems in the primary or in the secondary data center and the technical execution of the related procedures. The amount of unavailability of the service is the sum of the two time periods needed to complete above phases.

The Active-Passive (AP) configuration therefore entails the slowness of the system [5] reactivation procedures, while during normal operation the processing resources of the secondary site are dormant. In order to overcome these limits, TIPS adopted a different topology, in which both data centers are simultaneously operable in a workload sharing logic (Active-Active topology in Figure 1).

The Active-Active model speeds up the service recovery phase since systems are always active in both sites and a simple reconfiguration is needed to manage a site fail. For this reason, TIPS can achieve an RTO of 15 minutes.[6] However, with only two sites this reconfiguration procedure cannot be fully automated as human intervention is always necessary to understand if one of the two sites is unavailable or if the two sites are mutually isolated (split brain condition)[7].

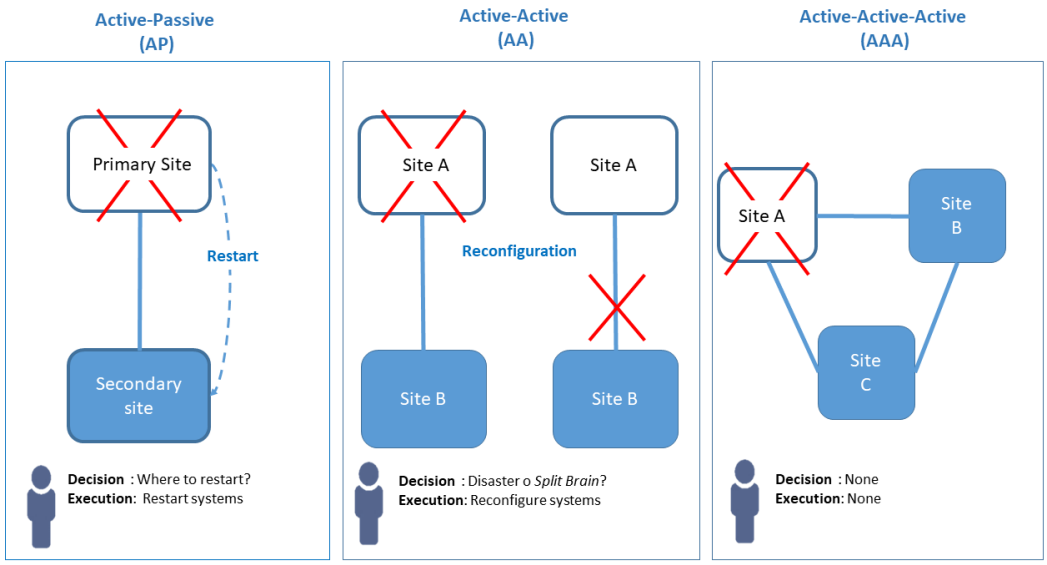


**Figure 1.** Business continuity models

Having an odd number of available processing sites (the minimum number of sites being obviously 3) enables the possibility to run consensus algorithms (see Consensus protocol box for details) among processing nodes. The nodes themselves are now able to automatically detect a site unavailability and then, independently from each other, activate a configuration change to converge towards a new topology which includes only the surviving sites. This model, shown in Figure 1, is known as Active-Active-Active and does not require any human intervention either in the decision phase or in the execution phase. TIPS is evolving towards this new approach.

5. Target2 and Target2 Securities systems adopt an Active-Passive (AP) business continuity model and are designed to guarantee a Recovery Time Objective (RTO) equal to 1 hour in the event of loss of one data center in the operational region.

6. The technical procedure to reconfigure the system to operate using only one of the two data centers actually takes less than 15 minutes which is the contractual Service Level Agreement (SLA) for the RTO.

7. Data or availability inconsistencies originating from the maintenance of two separate data sets with overlap in scope (Coulouris, Dollimore and Kindberg 2001). This is also commonly referred to as a network partition. If a network partition occurs in a consistent system then one side or the other (or both) of the partition needs to stop responding to requests to maintain the consistency guarantee. If both sides continue to respond to read and write requests while they are unable to communicate with each other, they will diverge and become inconsistent. This state where both sides of the partition remain available is called split brain. (Skeen, Davidson S. and Garcia-Molina H. 1985).

A good architectural topology is essential to achieving zero downtime during outages, but having in place autonomic operation procedures is important as well. In fact, there are many threats that can lead to a service disruption (e.g. natural disasters, cyber attacks, human errors, malfunctions, misconfigurations) and, no matter the source of the problem, recovery will be smoother, easier and faster with reliable continuous operation procedures [8].

One of the Business Continuity Management (BCM) mantras is to regularly test the recovery procedures in order to assure that all the requirements are met. The testing processes must support RTO and Recovery Point Objective (RPO)[9] goals. TIPS has tight constraints in term of service recoverability, in particular RTO is currently 15 minutes and RPO is 0. This means that the service has to be restored in 15 minutes with no data loss. In order to meet these requirements an extensive use of automation is fundamental.

**Consensus protocols**

A distributed system is a set of processes or nodes that cooperate to achieve a common result. A set made up of $N$ of these processes is said to be resistant to $k$ failures if it is able to continue to operate consistently and therefore achieve the result against the fault of $k$ processes. In TIPS the goal is to update a single ordered log of settlement transactions in a consistent way and so at any point in time only one node is able to write the log acting as a *leader* and the remaining $N - 1$ nodes are *followers* replicating the data written by the leader.

In this context, the consensus protocol is the mechanism that allows the set of nodes to reach an agreement (consensus) on the next node acting as leader in the face of the fault of the current leader. Naturally, it is relatively easy to solve crash faults[a]. Algorithms used to solve this type of faults are called crash fault tolerance (CFT) algorithms or non-Byzantine fault tolerance algorithms. Byzantine faults[b] may cause unauthorized changes, have higher complexity and are more difficult to solve. Generally, if a system is in a reliable internal network, only the issue of crash fault tolerance (CFT) is to be considered because all nodes are trusted and controlled by the same authority. TIPS is a CFT system.

In a CFT system that is resistant to a number of faults equal to $k$ and therefore is able to elect a new leader in a deterministic way, a number of nodes $N \geq 2k + 1$ is required.

*a*. Crash faults in a distributed network are essentially inevitable and are caused by unreliable and unstable physical hardware. For example, networks, or communication channels, cannot always be stable and reliable. Disks on physical machines or CPUs will not always be in good condition.

*b*. Malicious nodes in networks may change and forge data at any time, making it harder to solve the consensus problem. These troublemaking problems that may change and forge data or response information are often referred to as Byzantine faults.(Lamport, Shostak and Pease 1982)

The continuous operation strategy follows a generalised approach: regardless the number and type of the events occurred, the application is resumed at the same starting point. The unavailability of the primary site is the main assumption, so it is necessary to isolate this site following the relocation of all the services to the surviving site. This approach is led by the 2 data centers design. In particular, TIPS is designed with a distributed shared nothing architecture(Arcese, D. Di Giulio and V. Lasorella 2018) that relies on the duplication of processing nodes and the repetitive parallel processing on the same data stream. The aim of this design is to maximize the probability of always having a valid data stream in the face of multiple servers failure or even complete data center loss.

Therefore, concerning this part of the architecture, nothing should be done, the application

8. Please note that, in order to manage cyber attack scenarios, additional continuous operation procedures are required (e.g. in case of ransomware, data restore needs to be properly managed).

9. Recovery Point Objective (RPO) generally refers to the amount of data that can be lost within a period most relevant to a business, before significant harm occurs, from the point of a critical event to the most recent backup.

is resilient enough to overcome a failure. Anyway, some technologies (i.e. message brokers and database clusters) are not designed with this concept in mind and need some additional work. These technologies rely on a cluster stretched over 2 sites: if a disaster crops up, the cluster must be resized, meaning that the new cluster will be composed by only the surviving nodes. So, the 'business as usual' situation is shown in Figure 2:

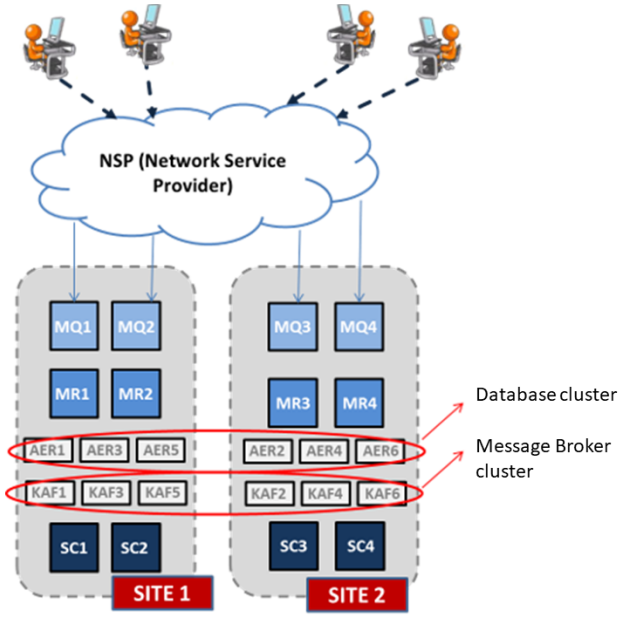


**Figure 2.** TIPS architecture

When a disaster crops up in SITE 1, all the clusters in SITE 2 have to be reconfigured. Although, as stated before, MQs, Message Routers (MRs) and Settlement Cores (SC) are independent and do not need any intervention, anyway a reconfiguration is needed for the other clusters (i.e. database and message broker). This is an inevitable approach with strict consistency requirements and a two data center architecture. So, in the event of a disaster, a service outage resulting from the decision time and the above mentioned reconfiguration must be accepted.

For this reason, a new architecture with 3 data centers has been investigated, as represented in Figure 3. Each data center in the diagram corresponds to a physical rack, moreover the chosen replication factor is 3. The 'rack awareness' feature of the database cluster will automatically spread the records among the servers inside the 3 racks; in detail, each rack (i.e. each site) will have exactly one copy of all the records. The clients can read either from the master records, which are distributed uniformly across the 3 racks, or from the local version, be it master or replica ("relaxed read approach"). The diagram in Figure 3 shows the "relaxed read approach" which optimizes read latency.[10]

10. For more details please see https://aerospike.com/blog/tie-breaker-functionality-for-aerospike-multi-site-clustering/

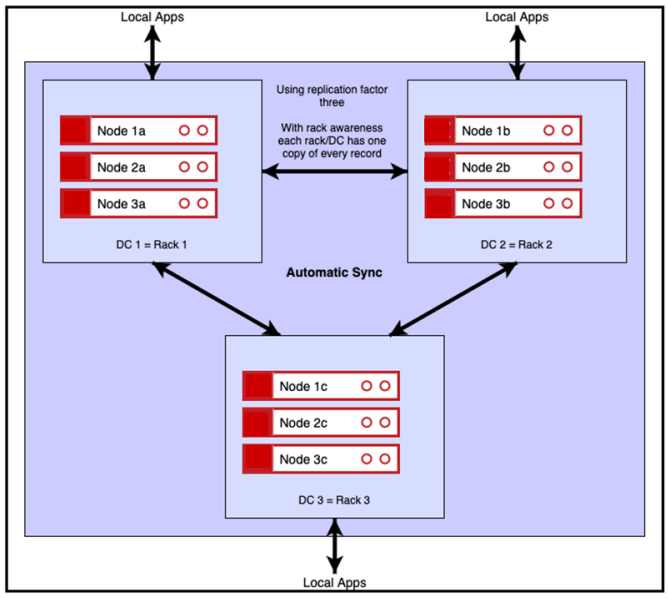


**Figure 3.** Database cluster configuration using 3 data centers: automatic fail-over

There will be no data loss if one of the three data centers becomes unavailable, because each of the 3 sites has a full copy of the data. The resiliency features of the cluster, in the event of a failure, will automatically redirect the reading and writing operations to the available data centers. The new configuration will guarantee an RTO equal to 0, without having to perform any fail-over procedure and allowing periodic disaster recovery tests with no service disruption.

## 2.1 Reliability and Availability

The reliability of a computer system is one of the main design parameters to be taken into account for always-on systems such as TIPS, involving lower operating risks and costs in the running phase.

The reliability (Weick 1989) is the probability that a system, including hardware, firmware, and software, will satisfactorily perform the task for which it was designed or intended, for a specified time and in a specified environment. In mathematical terms, the reliability of a single component is given by the function:

$$R(t) = e^{-\lambda t} = e^{-\left(\frac{t}{MTBF}\right)} \tag{1}$$

where $\lambda$ is the failure rate and $MTBF$[11] is the mean time between failures. (Johnson 1988)

System reliability is tightly coupled with system availability[12], which refers to the percentage of time that the infrastructure, system, or solution remains operational under normal circumstances in order to serve its intended purpose. In mathematical terms the single component availability is:

$$A = \frac{MTBF}{MTBF + MTTR} \tag{2}$$

11. Mean time between failures ($MTBF = \frac{1}{\lambda}$) is the predicted elapsed time between inherent failures of a mechanical or electronic system, during normal system operation.

12. At a given time, t, the system will be operational if one of the following conditions is met:

- The system functioned properly from 0 to t, i.e., it never failed by time t. The probability of this happening is $R(t)$.
- The system functioned properly since the last repair at time $u$, $0 < u < t$. The probability of this condition is: $\int_0^t R(t-u)m(u)du$ where $m(u)$ is the renewal density function of the system.

Consequently, the point availability is the sum of the above two probabilities: $A(t) = R(t) + \int_0^t R(t-u)m(u)du$. The steady state availability of the system is the limit of the availability function as time tends to infinity. The steady state availability is the long-run or asymptotic availability and is given by: $A = A(\infty) = \lim_{t\to\infty} A(t)$. It can be measured in terms of $MTBF$ and $MTTR$ (Mean Time To Repair).

where MTTR is the mean time needed to repair the component.

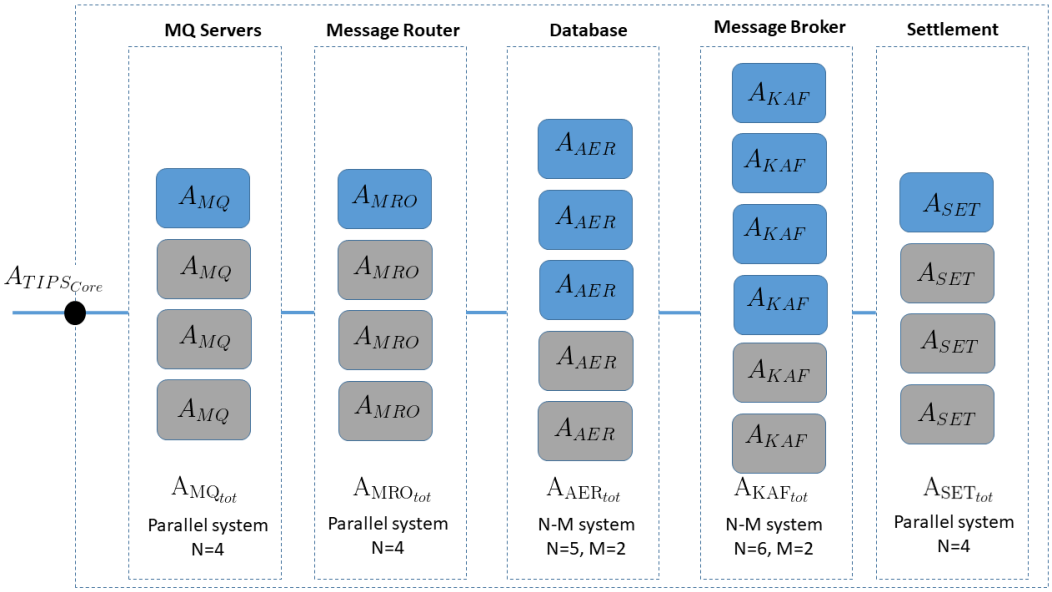


**Figure 4.** TIPS: Settlement engine availability model for the current 2 data center footprint (the correspondent reliability model is equivalent). The boxes highlighted in blue are the minimal set of servers needed to provide the service.

Starting from the reliability $R_i$ (or from the corresponding availability $A_i$) of any single component it is possible, through appropriate composition rules (see box System Availability composition for details), to derive the total reliability and the total availability of the system. In order to apply this process to the portion of the TIPS system devoted to settling payments (i.e. TIPS Settlement engine) it is necessary to refer to the system reliability model shown in Figure 4.

The component groups of the MQ, Message Router and Settlement contain 4 servers each and operate in shared-nothing mode requiring at least one single component to be available for each group to provide the service. They are therefore components that work in parallel and the overall reliability of each of these groups, in the assumption that they are statistically independent, can be calculated as: (See box System reliability composition rules eq.(7))

$$\begin{aligned} R_{MQ_{tot}} &= 1-(1-R_{MQ})^4 \\ R_{MRO_{tot}} &= 1-(1-R_{MRO})^4 \\ R_{SET_{tot}} &= 1-(1-R_{SET})^4 \end{aligned} \tag{3}$$

The database and message broker clusters are composed respectively of 5 and 6 nodes of which, however, at least 3 database nodes and 4 message broker nodes are always required. The (8) are applied to these clusters and then the corresponding overall reliability is calculated. It is:

$$\begin{aligned} R_{DB_{tot}} &= \sum_{k=0}^{2}\binom{5}{k}R_{DB}^{(5-k)}(1-R_{DB})^k \\ R_{MBR_{tot}} &= \sum_{k=0}^{2}\binom{6}{k}R_{MBR}^{(6-k)}(1-R_{MBR})^k \end{aligned} \tag{4}$$

In order for the TIPS settlement engine to be available, all the subsystems that compose it, must be

available and therefore by applying (6) the following total reliability and availability are obtained:

$$\begin{aligned} R_{TIPS_{Engine}} &= R_{MQ_{tot}} \cdot R_{MRO_{tot}} \cdot R_{DB_{tot}} \cdot R_{MBR_{tot}} \cdot R_{SET_{tot}} \\ A_{TIPS_{Engine}} &= A_{MQ_{tot}} \cdot A_{MRO_{tot}} \cdot A_{DB_{tot}} \cdot A_{MBR_{tot}} \cdot A_{SET_{tot}} \end{aligned} \tag{5}$$

**System reliability composition rules**

In a computer system made up of $N$ parts, the overall system reliability is calculated starting from the reliability $R_i$ of its individual components. These can be connected in series, in parallel or in more complex ways. To summarise, the following basic configurations are obtained:

- **Serial**: in a serial system composed by $N$ elements, each one is required to function properly for the system to function correctly. The corresponding reliability is the product of the reliability of each component:

$$R_{series} = \prod_{i=1}^{N} R_i \tag{6}$$

  i.e. if only one element fails, the reliability of the whole system is 0.
- **Parallel**: in a parallel system composed by $N$ elements, it is sufficient for one of them to be operational for the system to function correctly. The corresponding reliability is equal to one minus the probability that all the elements are failed:

$$R_{parallel} = 1 - \prod_{i=1}^{N} (1 - R_i) \tag{7}$$

- **N,M system**: in an $N$, $M$ system only $M$ out of $N$ components are required to function correctly for the system to function correctly. (Traylor 2016) The corresponding reliability is (assuming $R_i = R$ for each component):

$$R_{N,M} = \sum_{k=0}^{M} \binom{N}{k} R^{(N-k)} (1 - R)^k \tag{8}$$

By making reasonable assumptions on the frequency and the resolution time of faults, we can to use (5) to estimate the theoretical availability of the TIPS settlement engine. It is therefore considered a severe fault that makes a processing node totally unavailable and that a time of 4 hours (MTTR = 4 hours) is required to restore its correct functioning. In the assumption that a failure occurs once a month (MTBF = 720 hours) and the fault affects all types of components in the same way, it can be considered that $A_{MQ} = A_{MRO} = A_{DB} = A_{MBR} = A_{SET} = A = 0.9945$. Using equation (5) a total TIPS engine availability of 0.999994 is obtained which corresponds to an estimated service outage of $\simeq$ 3 minutes per year. This value, which is defined $A_{TIPS}^{Normal}$, represents the expected theoretical availability of the TIPS settlement engine in its normal configuration using two data centers in AA mode (see Figure 1).

In this configuration, it may be necessary to reconfigure the system to operate with just one data center (see Figure 5). This occurs for example in the case of extraordinary maintenance activities with a wide impact on the infrastructure of a site. In this configuration, that is defined as *Recovery*, only a subset of the servers are available[13] and the availability of the TIPS engine is $A_{TIPS}^{Recovey} = 0.9997$ which corresponds to $\simeq$ 150 minutes per year.

13. Compared to Figure 4 in a *Recovery* scenario only 2 MQ , 2 Message Router, 3 Database, 3 Message Brokers and 2 Settlement servers are available.

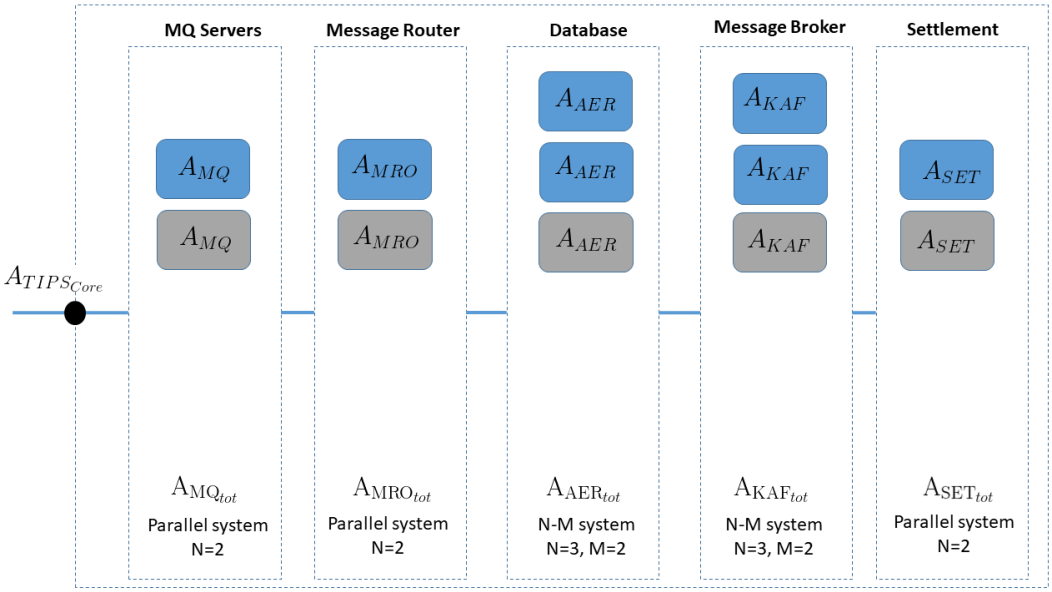


**Figure 5.** TIPS: Settlement engine availability model in a Recovery configuration. The boxes highlighted in blue represent the minimal set of servers needed to provide the service.

To further improve the system resilience it is necessary to adopt a three data center configuration in Active-Active-Active mode (see Figure 6). In this configuration[14] the total TIPS engine availability is 0.9999997 which corresponds to an estimated service outage of $\simeq$ 10 seconds per year.

So the complete unavailability of one site allows, in the "Recovery" configuration, a total availability of $A_{TIPS}^{Recovery} = 0.99997$ which corresponds to 15 minutes of service outage per year. This value is lower by a factor of 10 than the value (150 minutes) obtained in the two data center configuration.

14. In this configuration the number of MQ, Message Routers and Database servers is increased linearly while the number of Message broker servers remains the same and the number of settlement servers decreases by one. Reducing the number of settlement servers is the optimal choice for duplicate management, since each settlement server processes all payments and then each of them performs a settlement operation that generates a notification. All notifications must then be processed to eliminate duplicates and provide the user with a single receipt.

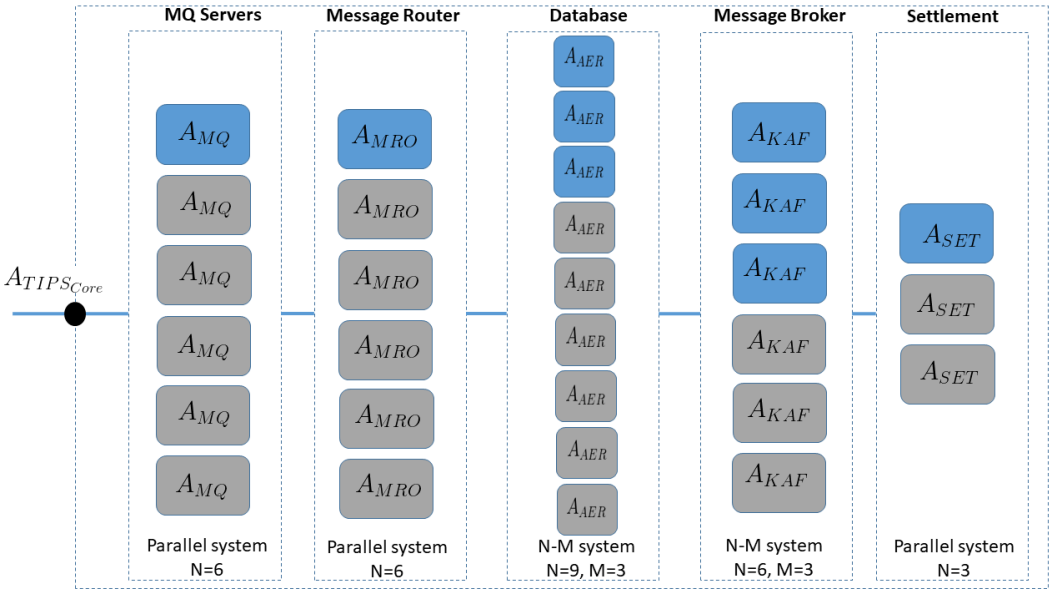


**Figure 6.** TIPS: Settlement engine availability model for the future 3 data centers footprint (the correspondent reliability model is equivalent). The boxes highlighted in blue represent the minimal set of servers needed to provide the service.

# 3 Automation

Automation of human activities in data centers was introduced more than 20 years ago, together with the execution of command lists and assisted command entry; since then, substantial improvements havw been achieved in all operational activities, from the opening/closing of day operations to the recovery/restart of ICT services, events management, end-to-end checks, etc.

A revolution in automation, rather than an evolution, has been achieved to manage TIPS, in the following sectors:

- service development, to reduce the time to market and increase the service quality ;
- service operations, to minimize the human effort and the running costs .

The automation of non functional tests in both the development and the operation phase helps to speed up the execution of the tests, making them repeatable over time and substantially contributing to the increase in the quality of ICT products.

Automation can also be used to support the operation of a 24H/7D ICT service as TIPS, in order to find solutions for autonomic data center management (Capotosto, Orsini and Tiberi 2017). In particular, automation can be effectively applied in specific domains to perform:

1. continuous integration and continuous delivery;
2. proactive monitoring and automatic remediation;
3. dynamic capacity management;
4. Non Functional Tests.

## 3.1 Continuous integration and continuous delivery

Continuous integration (CI) (Shahin, Ali Babar and Zhu 2017) (Soni 2015) refers to the building and integration stage of the software release process; it aims at regularly merging the code changes into a central repository, after which automated builds and tests are run. The key goals of continuous integration are to find and address bugs more quickly, improve software quality and reduce the time required to validate and release new software.

Continuous delivery (CD) (Häkli 2016) (Chen 2015) ensures that the software is reliably released at

any time. Over the years, getting software released has been a risky and time consuming process. The deployment process[15] has been based on the Maintenance Window (MW) concept, so any deployments should have been carried out within dedicated MW in which the service is closed according to a predefined Service Level Agreement.
Managing services available 24H/7D (i.e. no maintenance windows) has huge consequences on both the organizational and the technical side. Automation is still a major driver in reducing human effort and errors to a minimum, but it is not enough. In 24H/7D services, high availability and resiliency capabilities are ensured starting from the design phase with the introduction, among others, of the redundancy principle (e.g. considering a set of available processing nodes operating simultaneously in parallel). The deployment policy should also be carefully analysed to cope with the peculiar non functional requirements of the ICT service; as an example, a deployment strategy that applies the changes in parallel on all nodes (i.e. highlander deployment) cannot be considered a good choice, because it may be disruptive in terms of service availability.
A common solution, in a 24H/7D service scenario, is the rolling strategy deployment (Chaitanya 2020). In this case, the application is designed to handle two different software versions at the same time (the current version $N$ and the new version $N + 1$). By leveraging this requirement, the deployment process must be implemented following a sequential release logic (one node at a time) and must have control steps enabling the upgrade to the next node only after the successfully checking of the current node. The backwards compatibility is the main issue for developers: version N+1 must be completely compatible with version N as well as able to support and activate new functions.

Continuous Delivery concepts are used to help make the deployment process lean and agile. The goal of CD is to enable a constant flow of changes into production via an automated software production line. The CD pipeline is what makes it all happen.

A typical CI/CD pipeline includes the following main steps: build automation, automated test suites and merge capabilities in the CI part; automated deployment with certification tests and rollback capabilities for the CD part.

## 3.2 Proactive monitoring and automatic remediation

Proactive monitoring aims at anticipating events before they occur, as well as at being prepared and ready for an incident, if it happens. Monitoring shall be applied at two different levels: infrastructure (systems, storage and network equipment) and applications to provide a top-down view into the health status of the ICT services.

The infrastructure layer is in charge of verifying the utilization of resources like virtual CPUs (vCPUs), memory, I/O and system performances, such as indicators on queue length, shortage (or unavailability) of a resource, network transit time exceeded, etc. The application layer is in charge of verifying the presence of all modules (keep alive) as well as of continuously checking, with an end-to-end approach, if the whole architecture is behaving as expected.

Starting from the metrics detected by the monitoring system, actions aimed at anomalies resolution are then automatically activated. These actions are selected with suitable algorithms starting from a catalog of predefined actions.

## 3.3 Dynamic Capacity Management

Capacity Management has the task of providing IT capacity corresponding to both current and short term needs balanced against justifiable costs (Alasaly and Sanad 2016). In order to administer

15. Software delivery is the whole process of getting a software in production. It starts with conceptualization, continues with development and storing the code into the artifact repository, up to the deployment and installation into the server. So, the deployment is the last part of a delivery process, that makes a software available for use.

the current usage of services and components and to plan the capacity of the infrastructure, a Capacity Management Information System should be used and a capacity plan should be created. Efficient capacity management for ICT services is a challenging task, since the demand of any ICT service has high peaks, recurring and shifting seasonal patterns as well as bursts. Therefore, to ensure the respect of the SLA over time without over-provisioning, capacity management activities are required continuously. Traditionally, these activities were performed periodically, by a specific team in charge of making in-depth analysis on systems and workload performance historical data, analyzing trends, making forecasts and simulations including also the new services. The outcome is a capacity plan over an appropriate time-frame (2-3 years) and the correspondent investment needs. Moreover, the plan is then presented to and discussed with the company management, leading to a long and cumbersome process.

The process to put in place, instead, should automatically and continuously analyze all the available data, estimate trends, produce "what if" scenarios, to react, whenever possible, using automatic workflows (i.e. moving and optimizing ICT resource utilization). Its main objective should be to forecast the near future requirements and trends, in order to adjust the capacity plan and optimize the service performance and, in case of significant deviation from the capacity planning in place, raise warnings and alarms. Not all the scenarios can be covered dynamically; in fact, in case of new services or major releases that can require significant hardware or software improvements, the traditional approach (based on planned periodical activities) needs to be combined with the new one.

## 3.4 Non Functional Tests (NFTs)

Non Functional Tests (NFTs) can be automated to assess both the effectiveness (and the limits) of the above mentioned domains (i.e. CI/CD, dynamic capacity management, proactive monitoring and automatic remediation) and specific non functional "parameters", such as survivability and security.

NFTs play a substantial role in verifying the quality of the service (Capotosto, Orsini and Tiberi 2019). While functional requirements define what the system does, non functional requirements specify how the system should do it. The non functional test campaign assures that the application and configurations meet the non functional requirements of an ICT services throughout the application life cycle. A non functional test campaign of a 24H/7D ICT service should be executed when the service is up and running. For this reason, the test design has to guarantee measures to mitigate the risk of service unavailability or degradation, while continuing to assure the test effectiveness.

A non functional test campaign can aim at testing both the above mentioned automation domains and specific non functional parameters, which are never addressed by functional testing. In particular, although there are many non functional test parameters (for example: performance, resiliency, usability and reliability), this paper will focus on:

- **security**, measuring how a system is safeguarded against deliberate and sudden attacks from internal and external sources;
- **survivability** (Knight and Sullivan 2003), measuring if the software system continues to function and recovers itself in case of system failure.

Regarding the security parameter, a **vulnerability assessment** campaign is generally conducted. It is a review of the security posture of an information system, aimed at identifying and remediating known weaknesses and security misconfigurations. In particular, the vulnerability assessment evaluates if the system is compliant to a predefined configuration (usually based on recognized standards and best practices) or affected by known vulnerabilities.

Regarding survivability, two specific study cases can be explored: the continuous operation and the chaos monkey.

The **continuous operation** is a set of technical and management procedures that take place in the event of a business disruption. Continuous operations refers to an application running its business around the clock, regardless of external events or maintenance needs.

The **Chaos Monkey** works on the principle that the best way to avoid major failures is to fail constantly. However, unlike unexpected failures, which seem to occur at the worst possible times, Chaos Monkey has a configurable random schedule that allows simulated failures to occur at times when they can be closely monitored. In this way, it is possible to prepare for major unexpected errors rather than just wait for a catastrophe to strike and see how well it is managed. The final goal is building a predictive and repeatable operational model for any conditions and under any disturbances (e.g. multiple and simultaneous random hardware and software failures, high traffic conditions, unusual load profiles and so on).

### 3.4.1 TIPS automated test environment

In order to be able to verify the maturity of the tools and the organization supporting the data center management processes for around the clock services (24H/7D), a test strategy has been defined and specific roles and tools have been identified (Figure 7). In particular, the following roles have been involved:

- **Test Designer**, responsible for defining the scope of the test and preparing the technical specifications for its implementation;
- **Automation Developer**, responsible for implementing the automation scripts needed for the test execution;
- **Tester**, responsible for the test execution.

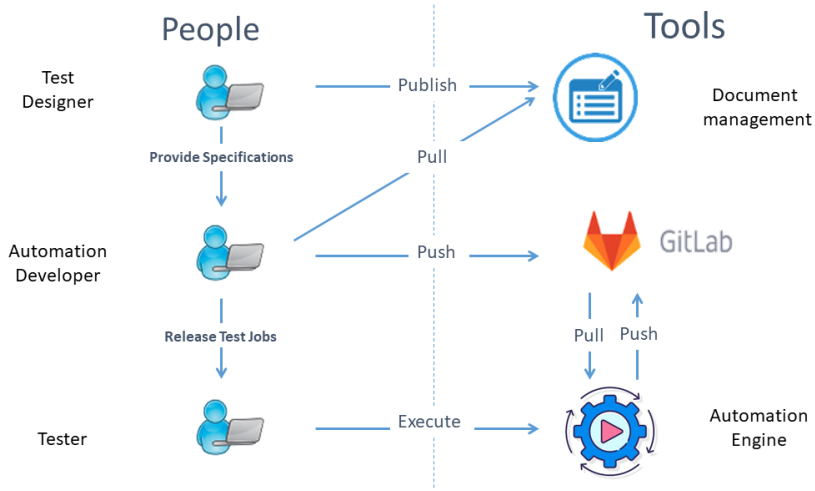


**Figure 7.** Typical test workflow

The Test Designer, in collaboration with the experts of the application and the infrastructure, defines the test perimeter and identifies the technical actions. The outcome of his work is a technical specification document that is published in the document management system. This document is used by the Automation Developer to build the automation scripts (playbooks published in a Git repository). The complete set of the NFT playbooks is available into the automation GUI web application. The Tester then accesses the automation GUI, executes the specific playbook and verifies its correct execution. Moreover the automation GUI allows the Tester to plan the test

execution in unattended mode and checks the relevant outcome later on. At the end of each test, all the related information is automatically collected and stored in the Git repository. When a test is launched or scheduled by the tester, the automation GUI invokes the automation engine module that is responsible for its execution. The automation engine can access, all Virtual Machines hosted on the Hypervisors in the two sites and is able to execute any action required by the playbook at Hypervisor, VM, OS or Application level (Figure 8).

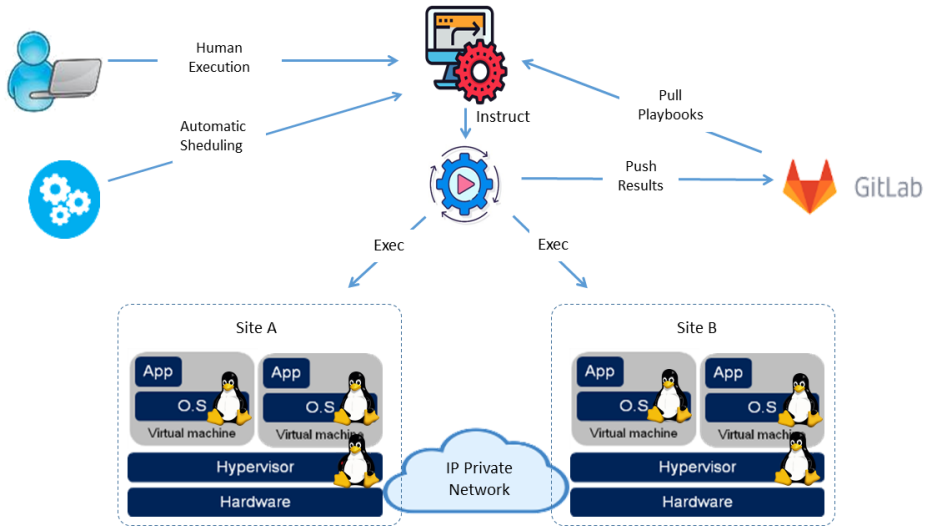


**Figure 8.** TIPS test execution workflow

However, it is necessary to carry out simulation tasks, represented with the SIMUL layer in figure 7, aimed at reproducing some operational scenarios useful to perform specific tests (to assess: Continuous Integration/Continuous Delivery, Proactive Monitoring, Dynamic Capacity Management, survivability and security). Simulation tasks are grouped into the following categories:

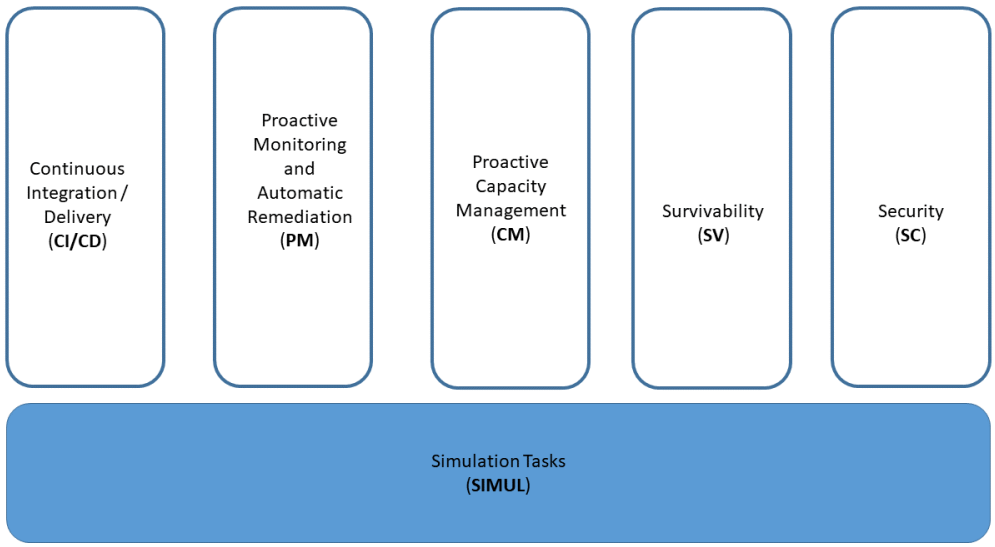


**Figure 9.** Test Architecture

- **Normal application behavior** and **Housekeeping activities**: this group includes scripts that simulate the normal user activity by injecting a configurable number of messages with a pre-defined length at a configurable rate (messages per second). Detailed statistics about the number of properly processed messages and the total transit time (end to end latency, see Figure 10) are collected for each test. The duration of each test is configurable (with a single parameter) and will be long enough to reach a stable condition. From now on the scripts from this group are referred to as SIMUL.APP.id or SIMUL.HOUSEKEEPING.id (where id is a numeric indicator to select the specific script in the group); typical housekeeping activities, needed to apply software changes, such as Kafka Broker rolling stop/start, are simulated with SIMUL.HOUSEKEEPING.id scripts too;
- **Error condition**: this group contains scripts that simulate an anomalous condition on one or more computing nodes (e.g. message brokers, database nodes). Some scripts labelled SIMUL.CPU.id create a specific CPU workload by executing demanding CPU bounded calculations involving one or more vCPUs of the computing node under test. SIMUL.MEMORY.id scripts allocate, in exclusive mode, a configurable amount of user space in order to simulate memory shortage conditions on the node under test.

Domain specific test activities are executed on top of one or more SIMUL scripts and the relevant statistics are analyzed in order to verify the effectiveness of Proactive Monitoring, Automatic Remediation, Continuous Delivery, dynamic Capacity Management and NFT tests.

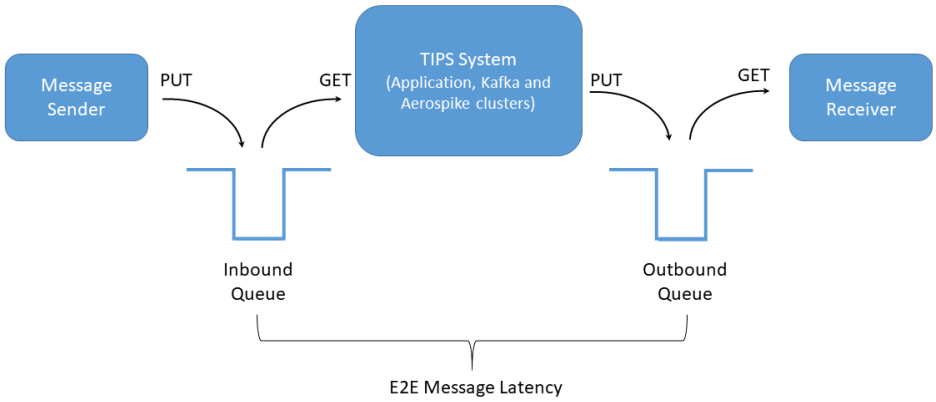


**Figure 10.** Reference architecture to calculate end to end message latency

### 3.4.2 Non functional test campaigns performed

**Continuous delivery** SIMUL.APP.01 script generates an application message with a timestamp and a constant payload of 5,120 bytes and sends 1,200,000 messages into an input queue with a constant rate of 1,000 messages per second (msg/s). Each message is processed by the settlement engine application and a new message is written into an output queue. This output message contains a new field called “latency” that is calculated by the application, using the input timestamp, taking into account the time needed for the message to pass from the input to the output queue. SIMUL.APP.02 is the same as SIMUL.APP.01 but with an increased message rate of 2,000 msg/s which is the upper performance boundary of the application under test while 1,000 msg/s is the nominal operating value. Furthermore, an infrastructural change is applied in order to verify the effectiveness of the deployment strategy. The related test is executed on all message brokers using the TEST.CD.INFRA.01 script, that performs the following actions on all message broker servers:

1. Stop Message Broker server;
2. Apply the patch;
3. Start Message Broker server.

Figure 11 shows the result of SIMUL.APP.01 + TEST.CD.INFRA.01: all the injected messages have been processed by the application with an average latency of 93 ms and a maximum latency of 9.061 ms. Figure 11 reports the message latency (1 second sampling). As clearly shown, the spikes correspond to the actions performed on the different message brokers (e.g. the spike around 13:02 am corresponds to the stop of the first message broker node). The amplitude and the duration of

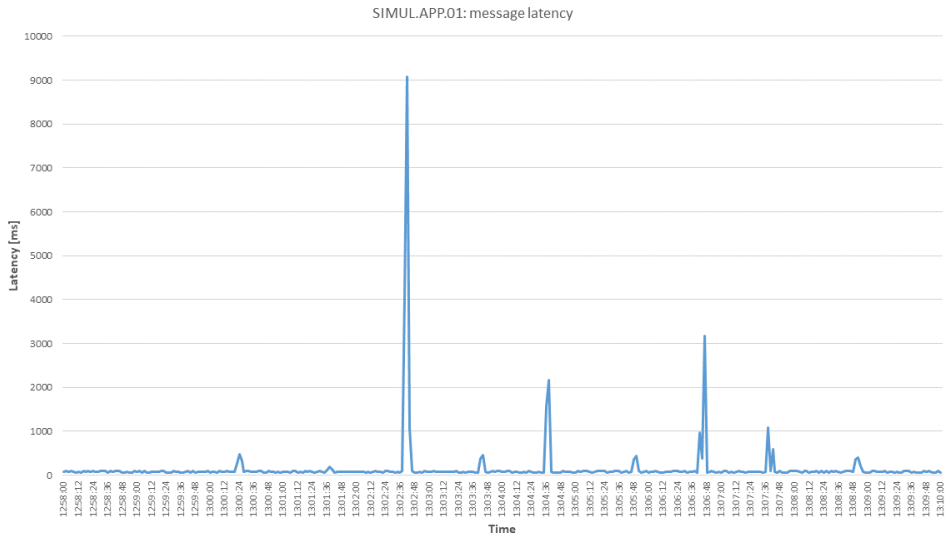


**Figure 11.** E2E message processing latency during execution of SIMUL.APP.01 : normal traffic conditions

the spikes can be explained taking into account the intrinsic characteristics of the message broker clusters and their interaction with the clients. One Input topic[16] with one single partition (i.e. one leader the clients connect to) and a replication factor equal to 3 (i.e. 2 additional internal replicas managed by 2 message brokers called followers) is defined. The Output topic is configured with 10 partitions (i.e. 10 leaders) and with the same replication factor of 3; this means that at any time

16. Message broker topics are the categories used to organize messages. Each topic has a name that is unique across the entire message broker cluster. Producers write data to specific topics, and consumers read data from specific topics. Message broker topics are multi-subscriber.Topics are partitioned, i.e. a topic is spread over a number of "buckets" located on different message brokers.

the system has 11 (1 input + 10 output) leaders to which the clients are connected, and 22 (2 input + 20 output) internal message broker replicas ready to serve the clients when a leader leaves the cluster. When the closure of a message broker node is forced, all clients that are connecting to this server, due to the fact that it is a partition leader, face a timeout and start a retry routine. The routine ends when one of the followers is elected as the new leader of the affected partition and so clients resume in normal state. This behaviour causes a spike in the end to end message latency. The results retrieved from SIMUL.APP.02+TEST.CD.INFRA.01 shown in Figure 11 and represent a high stress scenario for the application because SIMUL.APP.02 injects a number of messages per second (2,000 msg/s) comparable to the upper processing limit of the application under test.

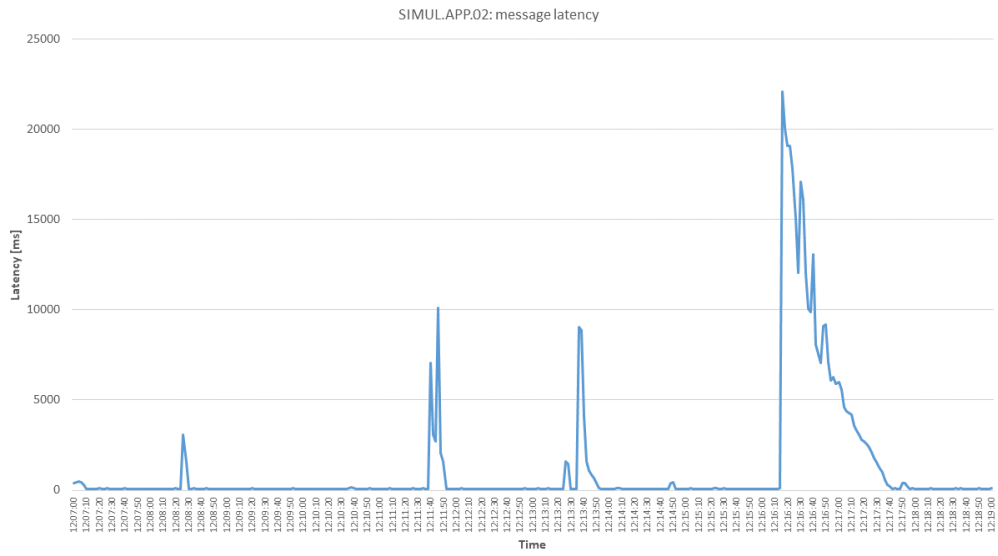


**Figure 12.** E2E message processing latency during execution of SIMUL.APP.02 : high traffic conditions

Regarding Figure 12, the shape of the curve is the same as that in Figure 11, but the magnitude of the impact of TEST.CD.INFRA.01 is amplified in terms of absolute maximum delay and duration of the perturbation. If a target SLA of 10 seconds for the message latency is set, while in Figure 11 this SLA is always respected in Figure 12 the SLA is breached in, at least, one interval.

Ultimately, the continuous delivery process described in the Continuous Delivery paragraph is suitable to be used in this scenario applying changes during low workload hours to prevent effects on application latency.

**Proactive monitoring and dynamic capacity management** SIMUL.APP.02 is used to evaluate the effectiveness of proactive monitoring. SIMUL.CPU.01 script runs on top of it; this allows the addittion of a significant CPU load on all nodes hosting the message processor used to read messages from the input queue and write messages to the output queue (see Figure 10). These nodes are configured with 2 vCPUs each and SIMUL.CPU.01 uses a tool to generate a real 100% CPU usage. The artificial 100% CPU workload is added in 2 distinct intervals: the first interval starts at T0 and ends at T1 (i.e. less than 2 minutes duration) while the second interval starts at T3 and lasts until the end of the test. TEST.PM.INFRA.01 is used to verify if the monitoring tool is able to detect the high CPU consumption and send a notification to the Automatic Remediation function in order to trigger an action to increase the number of vCPUs available to the message processor nodes. Figure 13 depicts the end to end message latency measured during TEST.PM.INFRA.01 execution. The normal average

latency is about 45 ms while during interval T0-T1 and T2-T3 the average latency increased to 90 ms as a direct result of high CPU consumption imposed by SIMUL.CPU.01. The proactive monitoring is configured to react to high CPU consumption lasting more than 2 minutes so in this case, during interval T0-T1 no action is performed. Then, at time T4 the proactive monitoring detects that the abnormal condition is lasting more than 2 minutes and reacts, triggering an increase CPU action via the automatic remediation function.

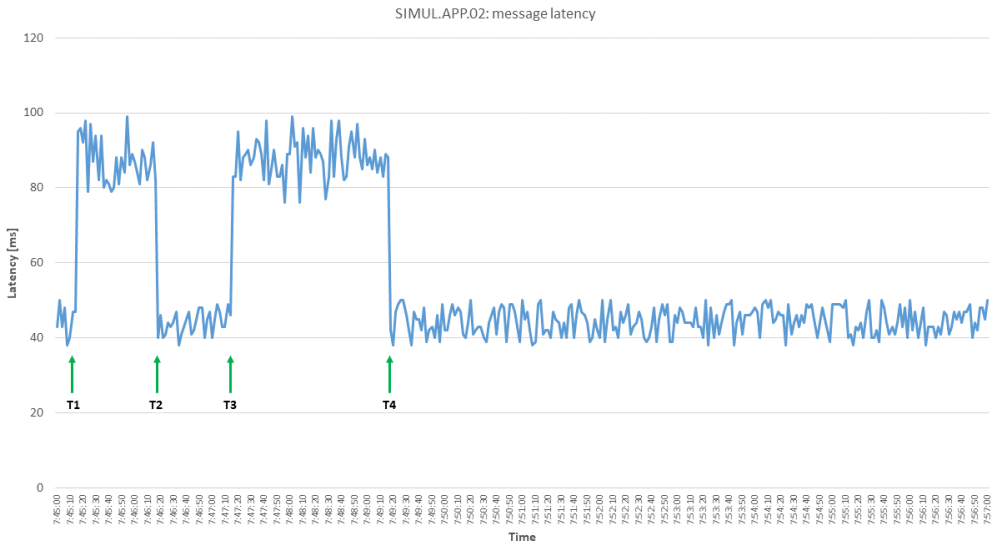


**Figure 13.** SIMUL.CPU.01

As a result of this action, 2 additional vCPUs are "hot added" to all message processor nodes and the end to end latency decreases to the normal value of 45 ms.

The results of the TEST.PM.INFRA.01 script represented in Figure 13 demonstrate that the combined action of Proactive Monitoring and Automatic Remediation can be very effective in addressing unexpected CPU workloads to quickly restore nominal application performances without any manual intervention.

**Survivability** As for survivability, two study cases are shown:

1. **Disaster Recovery in continuous operations**: it is a set of technical and management procedures that take place when a catastrophic event (earthquake, fire, flood etc.) makes the primary site completely unavailable. The goal of DR is to activate on a secondary site the mission critical applications with a valid copy of data in the minimum time (RTO) and with no data loss (RPO) ("ISO 22301:2019. Business continuity standard. Implementation guide" 2019)
2. **Chaos Monkey**: as already mentioned in paragraph 3.4 it entails randomly terminating instances in production to ensure that engineers implement their services to be resilient to instance failures.

The final goal is to build a predictive and repeatable operational model for any operating conditions and under any disturbances (e.g. multiple and simultaneous hardware and software failures, high traffic conditions, unusual load profiles and so on). Each NFT has a sequential structure as depicted in Figure 14. During the gathering facts phase system based information are collected (log, configuration files, process status, etc.). This phase ensures that the NFT starts from a well-defined system state. Phase 2 will be executed only if the NFT is performed in a test environment, where no

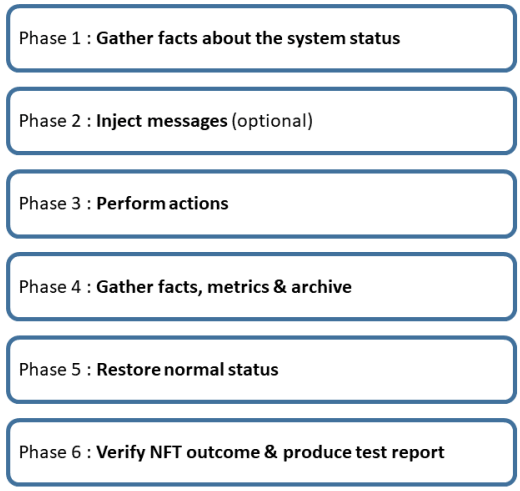


**Figure 14.** Survivability NFT execution phases

real traffic is flowing, in order to simulate production live conditions. It consists of a set of scripts that are able to send messages to the system and handle the relevant responses. In the test bed, TIPS application is used as an example of 24 H/7D system; an Instant Payment (pacs.008) is injected via the message sender application and the message receiver application is used to generate the relevant response in the form of a pacs.002 message (Figure 10). By using the message sender and the message receiver, the system can be stimulated in a way that is representative of the real message processing in the production environment.

Once the system is operating under real or simulated live conditions the NFT tests are ready to start. Phase 3 is the core of the activity and it is performed using automation playbooks that are a collection of scripts written in YAML language [17]. The automation scripts give the opportunity to use a lot of predefined modules to interact with virtual machines, network equipment, linux operating systems and applications in an easy and very efficient way requiring only a connection, without installing any agent on the system.

At the end of phase 3 another automation playbook is run, which gathers the results and archives them. In Phase 4 all systems are connected and all log files and metrics are collected. This allows the tester, to evaluate the outcome of the NFT (Phase 6). All information detected in this phase is packaged in a single archive file that is automatically stored in a GitLAB repository. The last automatic phase is to restore normal operation after the test is ended. The automation engine is used, in this environment, to deploy all the virtual machines, to install the software, and to configure all the infrastructure components (i.e. database cluster, message broker cluster, etc..).

Thanks to the idempotence[18] properties of automation scripts, the restoring of the normal status phase (Phase 5) is quite easy because it is only necessary to execute again the same automation playbook used to deploy the system. The automation engine restores all the software and the configuration to the original normal status.

Among the non functional tests this work focuses on 2 specific topics: the survivability test and

17. Yet Another Markup Language (YAML) is a human-readable data-serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted. YAML targets many of the same communications applications as Extensible Markup Language (XML) but has a minimal syntax.

18. Idempotence: the property of certain operations whereby they can be applied multiple times without changing the result beyond the initial application.

the security test. As concerns the survivability, two study cases are shown:

- disaster recovery test;
- Chaos Monkey test.

Regarding the security topic, the vulnerability assessment test campaign is presented.

**Disaster Recovery in Continuous operation:** The test is structured in this way: during phase1 the automation playbook automatically collects the configuration.

In this phase, all the necessary verification are performed to assure that an actual disaster condition is on-going, in order to prevent human errors or unwanted execution. If no disaster is in place and all the infrastructure is healthy, a message is shown and the playbook is stopped. In the second phase, 100 messages/sec are injected; this phase will be skipped during the test execution in the live production environment. Phase 3 is the actual test, in which the primary site is considered unavailable, a site isolation is performed and the playbook will resize the above mentioned clusters. Regarding the database and message broker clusters, the playbook resizes the clusters: (i) reducing the roster [19] of the database cluster to 3 nodes in SITE2 (see figure below); (ii) resizing the message broker cluster. After the playbook execution, the system is in the configuration depicted in Figure 15:

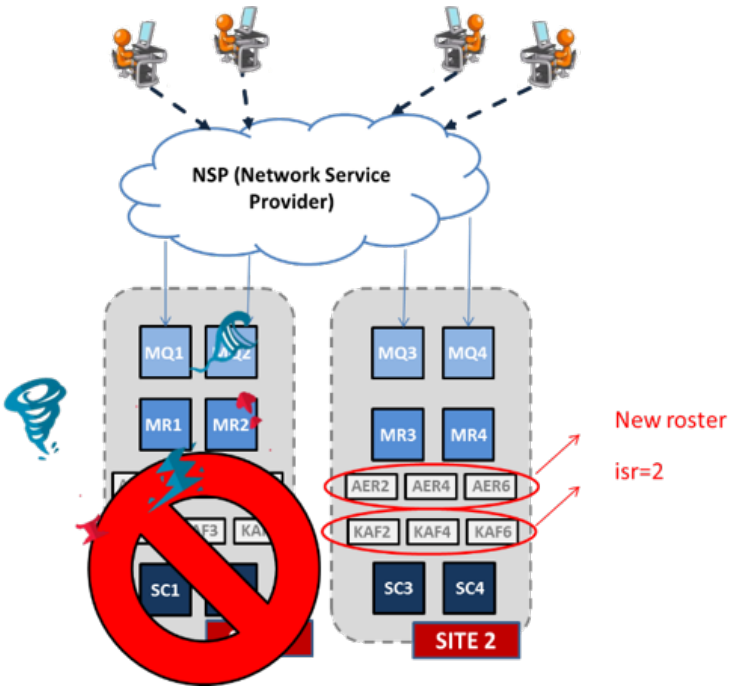


**Figure 15.** Disaster simulation

As stated before, in order the verify the correctness of the test itself, throughout the DR exercise, messages to the application are sent. This allows the verification that the data are processed before the disaster and after the failover is carried out. The theoretical situation described in Figure 16 should be met:

19. For the definition of the roster, please refer to the following: https://aerospike.com/products/features/strong-consistency/.

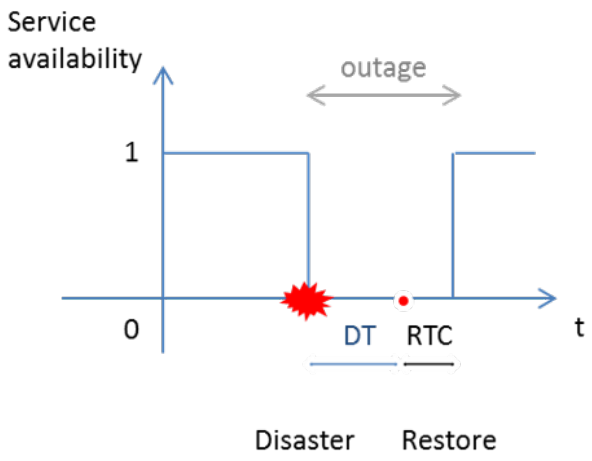


**Figure 16.** Service availability

During phase 4 the script checks the situation and assures that the infrastructure is in the failover condition. Whenever the playbook is ended, the business users are requested to verify that everything is up and running. The last phases are focused on collecting data in order to compare the expected behaviour of the system with the actual one, and then restore the 'business as usual' configuration and analyze the results. The application readiness is verified with the DR test execution. This playbook enables an automatic approach for the DR and ensures the system withstands a disaster as expected (assuring the SLA: RTO equals 15 minutes).

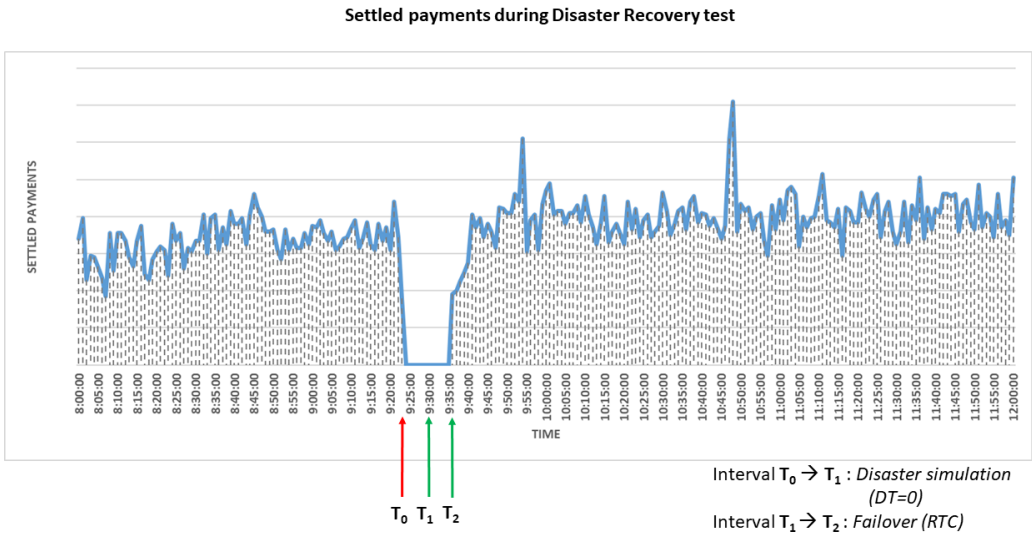


**Figure 17.** Settled payment during a TIPS DR test performed in the production environment

Figure17 shows the result of the TIPS production disaster recovery test performed on 5th March 2022. Interval $T_1 - T_0$ starting at 09:24 and ending at 09:30 includes the disaster simulation (network isolation of one site) and the decision time while the interval $T_2 - T_1$ starting at 09.30 and ending at 09.35 includes the failover activities to restore the service using the surviving site. As reported in the graph, the failover phase lasted 5 minutes and is well within the 15 minutes RTO.

**Chaos Monkey:** It is a complex, random test in which one or more combined failures take place. In this case a specific chaos monkey test has been designed to verify the response of TIPS. In particular the test targets the database and message broker cluster infrastructure in the test environment. This test should also be regularly performed in the live production environment. During phase1 all database and message broker server logs are collected and the size of the clusters is automatically retrieved via automation playbooks. In the test environment an injection of 100 messages/sec is performed during phase 2 of the test. This phase will be skipped during the test execution in the live production environment.

Phase 3 of this NFT is performed by shutting down randomly some database and broker nodes in order to reduce the cluster size to the minimum required to operate. The service should not be impacted by the reduction of the cluster size, according to the test objective. The database and broker node shutdown is performed by stopping the related Linux process. As depicted in Figure 18 the starting cluster size is 6 for both clusters and was changed several times during a 30 minute time frame, reaching in two intervals the minimum size required by the architecture design to operate normally. After each cluster downsize an automatic action tries to restore the desired cluster size (6 nodes). This action is triggered by the proactive monitoring system that detects the cluster size decrease and launches a remediation action. Thus, during the test execution, the NFT playbook tries to downsize the cluster while the proactive monitoring system tries to restore it (Figure 18). During the NFT execution the measured end to end message processing latency was always below the target SLA.

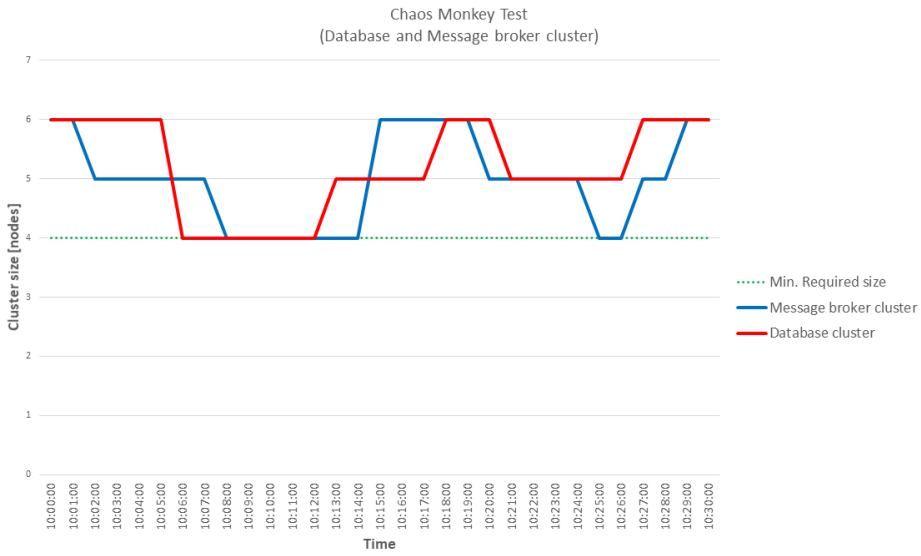


**Figure 18.** Database and message broker cluster size time evolution during the chaos monkey test

**Vulnerability assessment** Cyber threats can impact the ICT services availability and/or the data integrity or confidentiality. One of the measures useful to mitigate cyber risk is the vulnerability assessment process, aimed at periodically inspecting infrastructure components (e.g. servers, storage and network equipment) and applications (e.g. web applications) involved in the provision of ICT services, searching for technical software vulnerabilities and security misconfigurations. The reduction of exploitable weaknesses decreases the attack surface (and likelihood of impairments) and, at the same time, makes it harder for an attacker who has succeeded in breaching the perimeter defenses to achieve his or her objectives (e.g. by moving "laterally" or escalating privileges).

The vulnerability assessment is generally considered the less complex type of security assessment [20] ans is usually carried out (for verification and validation purposes) as part of a broader vulnerability and patch management (VPM) process.[21]

In TIPS environment, vulnerability assessments take place both periodically and on an event-driven basis,[22] and consist in three different phases: scan, analysis and remediation.[23]

In the first phase, vulnerability scans are performed with automated tools that inspect operating systems, middleware and applications, in order to identify known weaknesses.[24] These tools are based on distributed probes (deployed to reach specific subsets of targets), orchestrated remotely via APIs. This phase is executed during the release process, being a rolling deployment strategy adopted. In fact, it is useful to verify the absence of weaknesses in the new releases before their deployment, with the additional benefit of performing potentially harmful tests (which are usually not carried out on production systems).[25]

Afterwards, during the analysis phase, the vulnerabilities identified are verified in order to remove known false-positives and non-relevant weaknesses[26] prioritized following a risk-based approach[27]. Where appropriate and possible, information collected from external sources (e.g. commercial vulnerability feeds, public vulnerability databases) or provided by other internal processes (e.g. information sharing and cyber threat intelligence processes) are also considered in the analysis. Finally, in the remediation phase each weakness is addressed, engaging the team responsible for the activity and identifying an acceptable remediation time, consistently with the priority assigned in the previous step. If a vulnerability cannot be removed (e.g. because a patch does not exist yet, or because relevant impacts on the services are expected), alternative mitigation measures are applied (e.g. virtual patching [28]) and, if still present, the residual risk is evaluated and properly managed.[29]

20. Typical security assessments are, in order of complexity and depth of analysis: vulnerability assessment, penetration test, scenario based test and red team test.

21. The VPM can be considered a more complex process aimed at ensuring that technical (software) vulnerabilities are identified and eliminated, thus reducing potential risk situations and preserving confidentiality, integrity and availability of ICT services. Vulnerabilities can be identified both “indirectly” (referring to information available in the asset inventory) and “directly” (by inspecting systems with vulnerability scans).

22. For example, on-demand scans can be performed to quickly inspect critical systems searching for severe vulnerabilities that have been made public (i.e. 0 days).

23. If a broader VPM process exists, the analysis and the remediation phases can be fully delegated to such process.

24. The scans are conducted considering configuration baselines based on well recognized standards and best practices (e.g. CIS Benchmark - https://www.cisecurity.org/cis-benchmarks/).

25. Some vulnerabilities can be identified only, or more effectively, by performing tests that can affect system availability and/or data integrity and availability. Although these tests are usually avoided when assessing production systems, they can be executed with acceptable risks in a rolling deployment scenario.

26. Although some automation is used in this phase to search for vulnerabilities to be ignored (e.g. because already identified as not applicable or false-positive) and to consider information coming from internal cyber threat intelligence process (e.g. in CVSS evaluation), improvements are still ongoing to further reduce the human effort.

27. The severity of vulnerabilities is evaluated by using the Common Vulnerability Scoring System (CVSS), an open framework aimed at communicating the characteristics and severity of software vulnerabilities, owned and managed by FIRST.Org, Inc. (https://www.first.org/cvss). Whenever possible, the score is assigned automatically and using the Environmental metric group (refer to https://nvd.nist.gov/vuln-metrics/cvss for details).

28. Virtual patching (also known as “external patching” or “just in time patching”.) can be defined as “a security policy enforcement layer which prevents the exploitation of a known vulnerability” (https://owasp.org/www-community/Virtual_Patching_Best_Practices)

29. Residual risk, if not further mitigable, is usually accepted or, in extreme cases, avoided (e.g. by interrupting the

Therefore, it is evident how a proper use of the test architecture described in Chapter 3.4.1 can significantly improve the effectiveness of the vulnerability assessment process.

# 4 Conclusions

This paper examines two topics: i) service continuity and high availability architecture and ii) the importance of the automation and tests.

In order to cope with TIPS's strict non functional requirements (e.g. availability and service continuity) a new type of architectural design has been investigated. TIPS has been engineered with a high availability architecture, aimed at guaranteeing operational continuity in a zero downtime approach. Although there are no fixed rules, few good practices were available to gain the most out of the scarce resources. The new architecture identified allows the system, to withstand and recover from faults, making TIPS up and running 24/7/365 without any service interruption.

There are many features, patterns and configurations put in place to achieve this goal (i.e. data protection, recovery and replication, network load balancing, failover solution, geographical redundancy), but, among the others, this paper focuses on the benefits of clustering solutions for the availability, and consequently for the reliability of the service. Moreover, having an odd number of processing sites enables using consensus algorithms among the processing nodes. Regarding TIPS, the decision was to choose an Active-Active-Active configuration that does not require any human intervention either in the decision or in the execution phases.

Automation is the basis of the modern data centers and should be implemented throughout the ICT service lifecycle (implementation, transition and operations). Regarding the service implementation phase, the software engineering methodology has stressed the importance of automatic tests during coding. It is important to implement a test suite inside the continuous integration chain and to set-up a build system capable of executing arbitrary code during the CI pipeline. In this phase a collection of tests (unit, integration, smoke, functional, performance, security and so on) should be executed. This allows TIPS application to be less defective and to detect bugs at an early stage. Moreover, the application quality is increased thanks to the automatic code analysis (static and dynamic), the deployment algorithm (rolling update or canary deployment [30].) and the chaos engineering methodologies. Automation should also be applied in the next service lifecycle phases, to minimize the operational effort and the running costs. When the TIPS service is up and running, automation helps to reduce human errors, to increase activities repeatability and to improve the operation processes (monitoring, security and capacity planning and management). Referring to the major outcomes of the new architecture model described in this paper, the proactive monitoring detects all the anomalous events and triggers the corresponding automatic remediation process to run the correct workflow. The continuous deployment approach proved that the software can be released while the application runs. Moreover, the dynamic capacity management process enables the exploitation of the data collected, in order to predict trends. The automated approach applied to the NFT of a 24H/7D service showed significant benefits, minimizing human intervention and errors, increasing efficiency by offering the possibility to schedule the test at any time, without human presence. The automatic methodologies described in this paper can also be applied to improve cyber resilience capabilities: the inclusion of vulnerability assessments in the NFT tests, powered with increased automation, can help ensure that software vulnerabilities are quickly identified and properly managed, improving the timeliness and effectiveness of adaptation to evolving cyber threats. In addition, the automatic remediation concept can be extended to security

provision of the service).

30. A canary deployment is a deployment strategy that releases an application or service incrementally to a subset of users. All infrastructure in a target environment is progressively updated in small phases (e.g. 2%, 25%, 100%). A canary release is usually the lowest risk-prone of all deployment strategies, provided that a complete downtime is accepted for a limited amount of users.

incident response, introducing orchestration capabilities in order to simplify and speed up the reaction of incidents (in particular, during the containment and eradication phases).

## References


Alasaly, M., and A. Sanad. 2016. “Efficient Dynamic Capacity Management Approach for Guarantee SLAs in Clouds”. *TEM Journal* 5 (August). https://doi.org/10.18421/TEM53-07.

Arcese, M., D. Di Giulio and V. Lasorella. 2018. “Real-Time Gross Settlement systems: breaking the wall of scalability and high availability”. *Markets, Infrastructures, Payment Systems,* nos. 2021-2.

Capotosto, M., S. Orsini and P. Tiberi. 2017. “Continuous availability: from the shift paradigm to unmanned operations”. *CMG Impact 2017 annual conference.*

———. 2019. “Continuous availability: The role of non-functional testing in assuring continuous operations”. *CMG Impact 2019 annual conference.*

Chaitanya, R. 2020. “Comparison of zero downtime based deployment techniques in public cloud infrastructure”. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC),* 1082–1086. https://doi.org/10.1109/I-SMAC49090.2020.9243605.

Chen, L. 2015. “Continuous Delivery: Huge Benefits, but Challenges Too”. *IEEE Software* 32 (March). https://doi.org/10.1109/MS.2015.27.

Coulouris, G., J. Dollimore and T. Kindberg. 2001. *Distributed systems: concepts and design.* USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201619180.

Häkli, A. 2016. “Implementation of Continuous Delivery Systems”. PhD diss., (May). https://doi.org/10.13140/RG.2.2.33882.59846.

“ISO 22301:2019. Business continuity standard. Implementation guide”. 2019. *ISO Guide.*

Johnson, B. 1988. *Design and Analysis of Fault Tolerant Digital Systems.* USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201075709.

Knight, J., and K. Sullivan. 2003. “On The Definition Of Survivability” (May).

Lamport, L., R. Shostak and M. Pease. 1982. “The Byzantine generals problem”. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4 (3): 382–401.

“NIST SP 800-160 Vol. 2 Rev. 1”. n.d. *NIST.*

Renzetti, M., S. Bernardini, G. Marino, L. Mibelli, L. Ricciardi and G. Sabelli. 2021. “TIPS - TARGET Instant Payment Settlement Il sistema europeo per il regolamento dei pagamenti istantanei”. *Markets, Infrastructures, Payment Systems,* nos. 2021-01, https://www.bancaditalia.it/pubblicazioni/mercati-infrastrutture-e-sistemi-di-pagamento/questioni-istituzionali/2021-001/MIS-20210129.pdf.

Shahin, M., M. Ali Babar and L. Zhu. 2017. “Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices”. *IEEE Access* PP (March). https://doi.org/10.1109/ACCESS.2017.2685629.

Skeen, D., Davidson S. and Garcia-Molina H. 1985. “Consistency In A Partitioned Network: A Survey” (January).

“Survivable Network Systems: An Emerging Discipline”. 1997. *Software Engineering Institute.*

Soni, M. 2015. “End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery”. *2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM),* 85–89.

Traylor, R. 2016. *Reliability of Series and Parallel Systems with Correlated Nodes.* arXiv: 1601.03630 [math.PR].

Weick, K. E. 1989. “Mental models of high reliability systems”. *Industrial Crisis Quarterly* 3 (2): 127–142. https://doi.org/10.1177/108602668900300203.

## Papers published in the 'Markets, Infrastructures, Payment Systems' series

n. 1 TIPS - TARGET Instant Payment Settlement – The Pan-European Infrastructure for the Settlement of Instant Paymentsi, *by Massimiliano Renzetti, Serena Bernardini, Giuseppe Marino, Luca Mibelli, Laura Ricciardi and Giovanni M. Sabelli* (Institutional Issues)

n. 2 Real-Time Gross Settlement systems: breaking the wall of scalability and high availability, *by Mauro Arcese, Domenico Di Giulio and Vitangelo Lasorella* (Research Papers)

n. 3 Green Bonds: the Sovereign Issuers' Perspective, *by Raffaele Doronzo, Vittorio Siracusa and Stefano Antonelli* (Research Papers)

n. 4 T2S - TARGET2-Securities – The pan-European platform for the settlement of securities in central bank money, *by Cristina Mastropasqua, Alessandro Intonti, Michael Jennings, Clara Mandolini, Massimo Maniero, Stefano Vespucci and Diego Toma* (Institutional Issues)

n. 5 The carbon footprint of the Target Instant Payment Settlement (TIPS) system: a comparative analysis with Bitcoin and other infrastructures, *by Pietro Tiberi* (Research Papers)

n. 6 Proposal for a common categorisation of IT incidents, *by Autorité de Contrôle Prudentiel et de Résolution, Banca d'Italia, Commissione Nazionale per le Società e la Borsa, Deutsche Bundesbank, European Central Bank, Federal Reserve Board, Financial Conduct Authority, Ministero dell'Economia e delle Finanze, Prudential Regulation Authority, U.S. Treasury* (Institutional Issues)

n. 7 Inside the black box: tools for understanding cash circulation, *by Luca Baldo, Elisa Bonifacio, Marco Brandi, Michelina Lo Russo, Gianluca Maddaloni, Andrea Nobili, Giorgia Rocco, Gabriele Sene and Massimo Valentini* (Research Papers)

n. 8 The impact of the pandemic on the use of payment instruments in Italy, *by Guerino Ardizzi, Alessandro Gambini, Andrea Nobili, Emanuele Pimpini and Giorgia Rocco* (Research Papers) (in Italian)

n. 9 TARGET2 – The European system for large-value payments settlement, *by Paolo Bramini, Matteo Coletti, Francesco Di Stasio, Pierfrancesco Molina, Vittorio Schina and Massimo Valentini* (Institutional Issues) (in Italian)

n. 10 A digital euro: a contribution to the discussion on technical design choices, *by Emanuele Urbinati, Alessia Belsito, Daniele Cani, Angela Caporrini, Marco Capotosto, Simone Folino, Giuseppe Galano, Giancarlo Goretti, Gabriele Marcelli, Pietro Tiberi and Alessia Vita* (Institutional Issues)

n. 11 From SMP to PEPP: a further look at the risk endogeneity of the Central Bank, *by Marco Fruzzetti, Giulio Gariano, Gerardo Palazzo and Antonio Scalia* (Research Papers)

n. 12 TLTROs and collateral availability in Italy, *by Annino Agnes, Paola Antilici and Gianluca Mosconi* (Research Papers) (in Italian)

n. 13 Overview of central banks' in-house credit assessment systems in the euro area, by L*aura Auria, Markus Bingmer, Carlos Mateo Caicedo Graciano, Clémence Charavel, Sergio Gavilá, Alessandra Iannamorelli, Aviram Levy, Alfredo Maldonado, Florian Resch, Anna Maria Rossi and Stephan Sauer* (Institutional Issues)

n. 14 The strategic allocation and sustainability of central banks' investment, by *Davide Di Zio, Marco Fanari, Simone Letta, Tommaso Perez and Giovanni Secondin* (Research Papers) (in Italian)

n. 15 Climate and environmental risks: measuring the exposure of investments, by *Ivan Faiella*, *Enrico Bernardini*, *Johnny Di Giampaolo*, *Marco Fruzzetti*, *Simone Letta, Raffaele Loffredo and Davide Nasti* (Research Papers)

n. 16 Cross-Currency Settlement of Instant Payments in a Multi-Currency Clearing and Settlement Mechanism, by *Massimiliano Renzetti*, *Fabrizio Dinacci and Ann Börestam* (Research Papers)

n. 17 What's ahead for euro money market benchmarks?, by *Daniela Della Gatta* (Institutional Issues) (in Italian)

n. 18 Cyber resilience per la continuità di servizio del sistema finanziario, by Boris Giannetto and Antonino Fazio (Institutional Issues) (in Italian)

n. 19 Cross-Currency Settlement of Instant Payments in a Cross-Platform Context: a Proof of Concept, *by Massimiliano Renzetti, Andrea Dimartina, Riccardo Mancini, Giovanni Sabelli, Francesco Di Stasio, Carlo Palmers, Faisal Alhijawi, Erol Kaya, Christophe Piccarelle, Stuart Butler, Jwallant Vasani, Giancarlo Esposito, Alberto Tiberino and Manfredi Caracausi* (Research Papers)

n. 20 Flash crashes on sovereign bond markets – EU evidence, *by Antoine Bouveret, Martin Haferkorn, Gaetano Marseglia and Onofrio Panzarino* (Research Papers)

n. 21 Report on the payment attitudes of consumers in Italy: results from ECB surveys, *by Gabriele Coletti, Alberto Di Iorio, Emanuele Pimpini and Giorgia Rocco* (Institutional Issues)

n. 22 When financial innovation and sustainable finance meet: Sustainability-Linked Bonds, *by Paola Antilici, Gianluca Mosconi and Luigi Russo* (Institutional Issues) (in Italian)

n. 23 Business models and pricing strategies in the market for ATM withdrawals, *by Guerino Ardizzi and Massimiliano Cologgi* (Research Papers)

n. 24 Press news and social media in credit risk assessment: the experience of Banca d'Italia's In-house Credit Assessment System, *by Giulio Gariano and Gianluca Viggiano* (Research Papers)

n. 25 The bonfire of banknotes, *by Michele Manna* (Research Papers)

n. 26 Integrating DLTs with market infrastructures: analysis and proof-of-concept for secure DvP between TIPS and DLT platforms, *by Rosario La Rocca, Riccardo Mancini, Marco Benedetti, Matteo Caruso, Stefano Cossu, Giuseppe Galano, Simone Mancini, Gabriele Marcelli, Piero Martella, Matteo Nardelli and Ciro Oliviero* (Research Papers)

n. 27 Statistical and forecasting use of electronic payment transactions: collaboration between Bank of Italy and Istat, *by Guerino Ardizzi and Alessandra Righi* (Institutional Issues) (in Italian)