


High availability model of a web service from a system administrator's perspective

Robert Tomasz Wirski

 <https://orcid.org/0000-0002-1654-2109>

Koszalin University of Technology, Faculty of Electronics and Informatics
2 Śniadeckich St., 75-453 Koszalin, Poland
e-mail: robert.wirski@tu.koszalin.pl

Keywords: web service, cloud computing, high availability, high reliability, operating system, redundancy, single point of failure, algorithm

JEL Classification: L86

Abstract

Based on cloud providers' reports on service outages, it has become clear that how a web service is deployed is of great importance. Clearly, using one service supplier is insufficient because it introduces single points of failure. In this paper, a novel high-availability multi-cloud model intended for a web service is proposed, which is free from such shortcomings yet preserves convenient assets of computing clouds. The methodology used to improve web service availability should involve several cloud suppliers and devise management techniques that control access to them. This is achieved by means of the server availability tracking algorithm, which controls client apps' access to the service. Moreover, typical benefits and problems involved in choosing IT infrastructure for a web service are elaborated. State-of-the-art cloud computing models, such as IaaS, PaaS, SaaS, BPaaS, and INaaS, are outlined. Operating systems statistics used for web services are included. Open-source monitoring software solutions are gathered, which help administrators to monitor and govern web servers.

Introduction

In this paper, a web service is a combination of hardware (computers) and software that processes the information and facilitates data sharing through users running clients' apps. The term "system" is also used for a similar meaning, especially when it presents a wider context. Starting a web service is a complex problem involving multiple arrangements to be made and facing competing priorities. Depending on company size and internal organization, decision-making patterns may vary. Usually, they are based on one or a combination of the following:

- Personal preference of employed system administrators,
- Management decisions,
- Repetition of previously made decisions.

As there is no "best" solution, the result is a trade-off that impacts the service performance, reliability, and costs (Lang, Wiesche & Krcmar, 2018). Some common challenges are listed below, which should be considered when starting a web service.

The high availability (HA) of a system means that it is capable of performing the intended tasks in a premised way and time-of-operation (Atchison, 2020). Reliability in this context is a similar term – it guarantees the correctness of systems operations. It is the responsibility of the system designer to ensure its reliability. Nowadays, systems are often implemented using dedicated hosting services. In such a case, availability is taken up by a cloud provider that provides dedicated solutions for this matter, usually in the form of load balancers. They redistribute tasks over a set of available resources, like hardware or virtual servers (Figure 1).

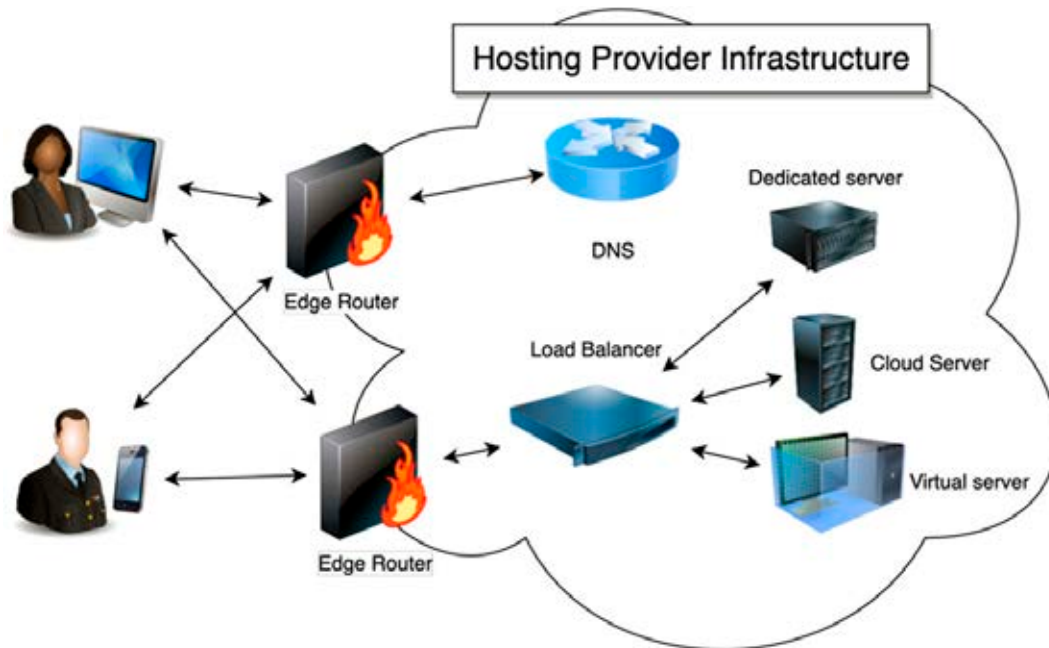


Figure 1. Distributing of incoming network traffic by means of a load balancer

Implementing the web service using local servers is especially reasonable if the company already runs its own server rooms with some free space and employs technical and IT administration staff. Here, the main costs would be the purchase of new hardware and energy. Unless it is just a single server rack, building a server room is both a costly and complex matter due to standards requirements (ANSI/BICSI, 2019), technology (Cisco, 2024), and meeting contradictory demands (Lowe, Green & Davis, 2016; Carapola, 2018; Atchison, 2020; Geng, 2021). There is also a space for scientific research in this area (Jadhav & Chaudhari, 2015; Ahmed, Bollen & Alvarez, 2021; Clement et al., 2023). The company is fully responsible for maintaining both hardware and software for their service. On the benefit side, such an approach does not depend on a third party, which eliminates risks related to service price changes or contract termination perturbations.

In the case of the dedicated servers hosting model, the company leases servers located in the host provider's server rooms connected to the internet with prearranged bandwidth. They can be accessed virtually via www panels, ssh, virtual desktops, etc. The hirer is fully responsible for installing and managing software. However, there are operating system images prepared to initiate the hosts, and the user is usually provided with a backup facility, typically 100 GB. The hosting provider is responsible for keeping the hardware up and running while providing tools to protect it from denial of service (DDoS) attacks. The advantage of such an approach is that

the company does not need to run its own server rooms. There is no hardware management and servicing, meaning that employment costs can be reduced to just IT administrators and, possibly, software developers. The disadvantage is a dependency on a third party. So, migration scenarios should be ready in case of problems with the suppliers. Because the company data resides in a third-party location, a problem with data security arises. It is especially important when personal or sensitive data is collected. A legal agreement with the host vendor is a must in this case.

Cloud hosting is a relatively new area of computing that is under active development by host providers. The definition of cloud computing according to the National Institute of Standards and Technology (Mell & Grance, 2011): "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

It is characterized by on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. Typically, cloud computing is classified into the following models:

- Infrastructure-as-a-service (IaaS),
 - Platform-as-a-service (PaaS),
 - Software-as-a-service (SaaS),
- which are further generalized into (Ruparelia, 2015):
- Process-as-a-service (BPaaS),
 - Information-as-a-service (INaaS).

In the IaaS model, a set of virtual raw IT resources is provided for the users who can develop their projects without local infrastructure. The resources are customizable and can be easily fitted to users' needs. In PaaS, developers are provided with a platform that can be used to develop software without the need to maintain a local hardware infrastructure. It is centralized, so collaboration between teams is easy to achieve. The SaaS model is for those who need software to meet their specific business needs. Companies choosing such a model are required to maintain internet access to their departments only and pay a chosen subscription fee. BPaaS is at an even greater level of abstraction, where no programming is required, and the cloud provider offers ready-to-use solutions for a specific business model. The INaaS model is relatively new, and some ambiguities can be observed. For example, a similar acronym is used for integration-as-a-service, indoors-navigation-as-a-service, and innovation-as-a-service. However, the term "information-as-a-service" seems to be the most widely accepted (Ruparelia, 2015). There are few commercial systems that claim to be built on this model (Table 1).

Table 1. Commercial examples of cloud computing models

Model acronym	Commercial examples of cloud computing models
IaaS	Amazon Web Services, Microsoft Azure, Google Compute Engine
PaaS	Google App Engine, Windows Azure, Adobe Commerce
SaaS	Google Apps, Microsoft Office 365, Dropbox, GitHub
BPaaS	eBay auction service, PayPal service
INaaS	BEA AquaLogic Data Services Platform

Recently, there has been interest in the multi-cloud or hybrid-cloud approach that involves two or more cloud providers (Petcu, 2013). Comparison of two research projects of PaaS multi-cloud architectures, namely ASCETiC (<http://www.ascetic-project.eu>) and SeaClouds (<http://seacLOUDS-project.eu>), is provided in previous work (Ferrer, Pérez & González, 2016). In other research (Sen et al., 2019; Ramamurthy et al., 2020), the authors discussed cost and timing optimization of resource selection in a multi-cloud environment. However, there are no papers known to the author that deal with security or high availability in multi-cloud models.

In the case of configuring local or dedicated hosts, an operating system selection problem arises.

In Figure 2, the popularity of Unix versus Windows is presented.

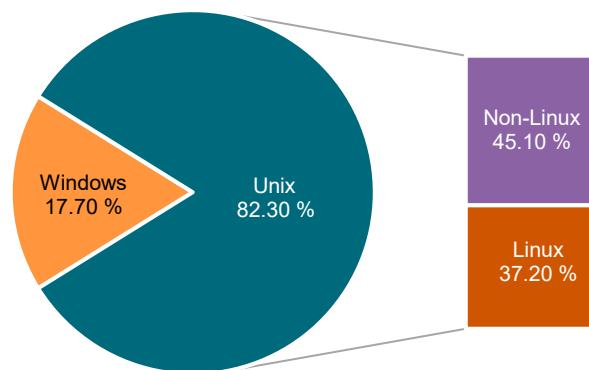


Figure 2. Report on operating systems usage for websites whose operating systems are known (W3Tech.com, 11 December 2023)

Clearly, Unix-like operating systems are more frequently chosen for websites compared to their Windows counterparts. Linux is used on 37.2 % of Unix-like platforms. The most popular Linux distros are Ubuntu, Debian, and CentOS (Figure 3). Interestingly, 42.6 % of them remain unknown.

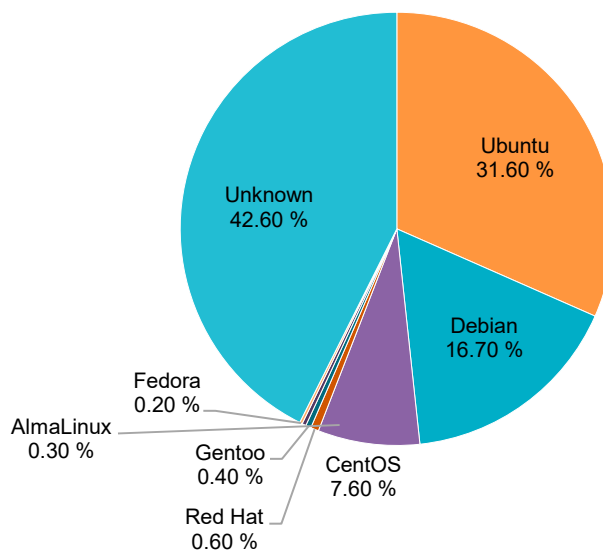


Figure 3. Linux distros popularity for websites (W3Techs.com, 8 December 2023)

Typical duties of system administrators who deal with security are:

- Monitoring network traffic,
- Monitoring system's hardware and software,
- Creating backups and verifying their correctness,
- Patching firmware and software.

To remove human factors from the process, a dedicated host managing software should be incorporated. Typically, its sales business model incorporates an open-source approach for a software core

expanded with more advanced modules and support based on a subscription basis. For example, the following monitoring software is based on this model:

- Nagios (www.nagios.org),
- Icinga (icinga.com),
- Zabbix (www.zabbix.com).

Several solutions exist to maintain server infrastructure:

- Kubernetes (kubernetes.io),
- Forman (theforeman.org), Puppet (puppet.com),
- Proxmox (proxmox.com).

Cloud providers have their own or adopted tools to support their clients in server orchestration, such as:

- Azure Automation from Microsoft,
- AWS CloudFormation from Amazon.

Any infrastructure is prone to failure. Even cloud providers encounter problems. In Table 2, reports on outages of selected providers have been collected.

Clearly, an administrator must consider web service availability problems during the system design and maintenance phases. In general, the host vendors provide their own solutions to maintain HA. In a typical HA model, shown in Figure 1, a client application connects to the provider's DNS to resolve a service's name into an IP address. When the IP has been acquired, the app starts sending requests to the hosts via load balancers. However, the presented model suffers from several single points of failures (SPOFs), which are:

- Edge routers,
- Load balancers,
- DNS servers.

Whenever any of the SPOF fails, the entire web service is down. The idea presented in this paper is to elaborate on a novel HA model that is free from

SPOFs without sacrificing the benefits of cloud providers.

Methods

To overcome the shortcomings of a typical HA model discussed in the Introduction, the author determined the following set of demands:

1. No SPOFs are allowed.
2. The service infrastructure needs to be scalable.
3. The web service should be able to work correctly even when only one cloud provider is active.
4. It should be possible to add another server to the service, even when some hosting providers are down.
5. When all infrastructure is down, the client app should wait for any server to return to operation and then restore the service gracefully (no app restart needed).

In the following section, an original author's conception of HA is presented that fulfills the demands listed above.

Results

In Figure 4, a novel HA model is presented. To fulfill demands no. 1 and 2 in the *Methods* section, at least two cloud suppliers need to be incorporated into the service. It must be tailored in such a way that only one cloud provider is enough to maintain the service under a typical load. Scalability using cloud resources or load balancing can also be used for this task. To achieve this, three extensions need to be implemented into the service:

1. Data synchronization between cloud providers,
2. Maintaining the system in case of malfunction,
3. A server availability tracking algorithm implemented in the client app.

Table 2. Cloud providers report on service outages from April to July 2023 (<https://isdown.app/blog/>)

Provider	April 2023		May 2023		June 2023		July 2023	
	Total incidents	Total outage	Total incidents	Total outage	Total incidents	Total outage	Total incidents	Total outage
		Time [h]		Time [h]		Time [h]		Time [h]
AWS	6	12.4	1	1.3	2	4.5	2	4.3
Azure	1	4.4	0	0	3	26.3	3	26.4
DigitalOcean	8	30.3	13	428.7	7	49.7	5	12.8
Fly.io	9	21.1	7	14.3	8	92.5	5	6.1
Heroku	9	1239.9	11	590.4	5	787.3	4	526.2
Linode	1	11.5	10	490.3	10	66.1	10	122.1
Netlify	1	0.3	0	0	7	8.4	3	30.3
Vercel	5	14.4	15	59.3	7	10.2	11	93.4

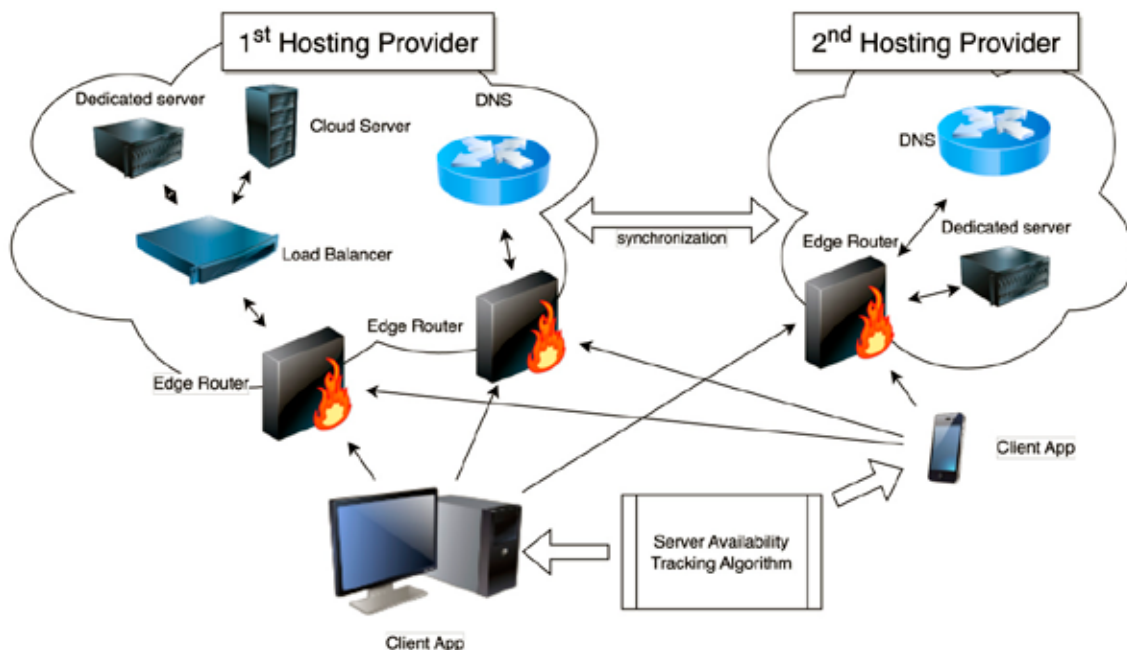


Figure 4. High availability redundant model for a web service

Data synchronization between cloud providers

To maintain integrity between hosts, the synchronization algorithm needs to be implemented. Depending on the operating system chosen for the hosts, possible approaches vary. In the case of Linux systems, standard Unix tools can be used for this task such as *rsync*. A single network file system (NFS) or similar, connected to all the cloud providers, should be avoided since it creates SPOFs. Disk synchronization between hosts can be performed intelligently only when data is changing, possibly with additional resynchronization once a day when the service load is the lowest. In the case of databases, standard replication mechanisms can be used. Another approach would be to write to all the databases on the fly when needed.

Maintaining system operation in case of malfunction

The malfunction of the service is defined as a situation when there are problems with the infrastructure of one or more of the cloud providers. To fulfill demand no. 3 in the *Methods* section, the client app needs to relocate to another working server. To achieve this, it must be equipped with a server availability tracking algorithm, which is described in the next subsection. The system malfunction also affects the data synchronization described above. Trying to communicate with a server that is down should be avoided because it generates timeouts, which additionally slows down an already flawed

service. Instead, a monitoring system should be implemented that informs the IT administrators about the problems.

Server availability tracking algorithm

In the proposed HA model, resources are scattered throughout several cloud suppliers. So, to make them usable for the service, the client app must track the availability of the servers, possibly measuring a transfer speed, and decide which one to use. In Figures 5–7, the algorithm suited for this task is outlined. It utilizes variables collected in Table 3. The object *xhr* used here originates from *XMLHttpRequest* API (https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest_API). The procedure should be initialized as follows:

```
xhr.onload = svrResponse;
xhr.onreadystatechange = svrMonitor;
xhr.onerror = svrError;
scanServers();
```

The algorithm is divided into four procedures due to the asynchronous nature of the *XMLHttpRequest* servers' communication API. In *scanServers()*, the number of active servers and the status of the service are evaluated using *active_servers* and *status* variables, respectively. At the end of the procedure, a call to a currently selected server is initiated via *xhr.open()*. When the server response arrives, *svrResponse()* is executed, and, if it is correct, the server is considered active. It is also a suitable place to receive the data of the service. If there are communication

Table 3. Variables used in server availability tracking algorithm

Name	Type	Description
<i>servers_length</i>	integer	number of structures in <i>servers</i>
<i>server_index</i>	integer	index of the currently tested server
<i>active_servers</i>	integer	number of servers with <i>active</i> variable set to true
<i>servers[0,...,server_length-1]</i>	struct {string: name, integer: tcp_port, boolean: active}	list of all servers assigned for the service
<i>status</i>	enumeration {OK, WARNING, CRITICAL}	status of the internet service
<i>xhr</i>	object with members: open, send, onload, onerror, onreadystatechange, status, readyState	servers' asynchronous communication

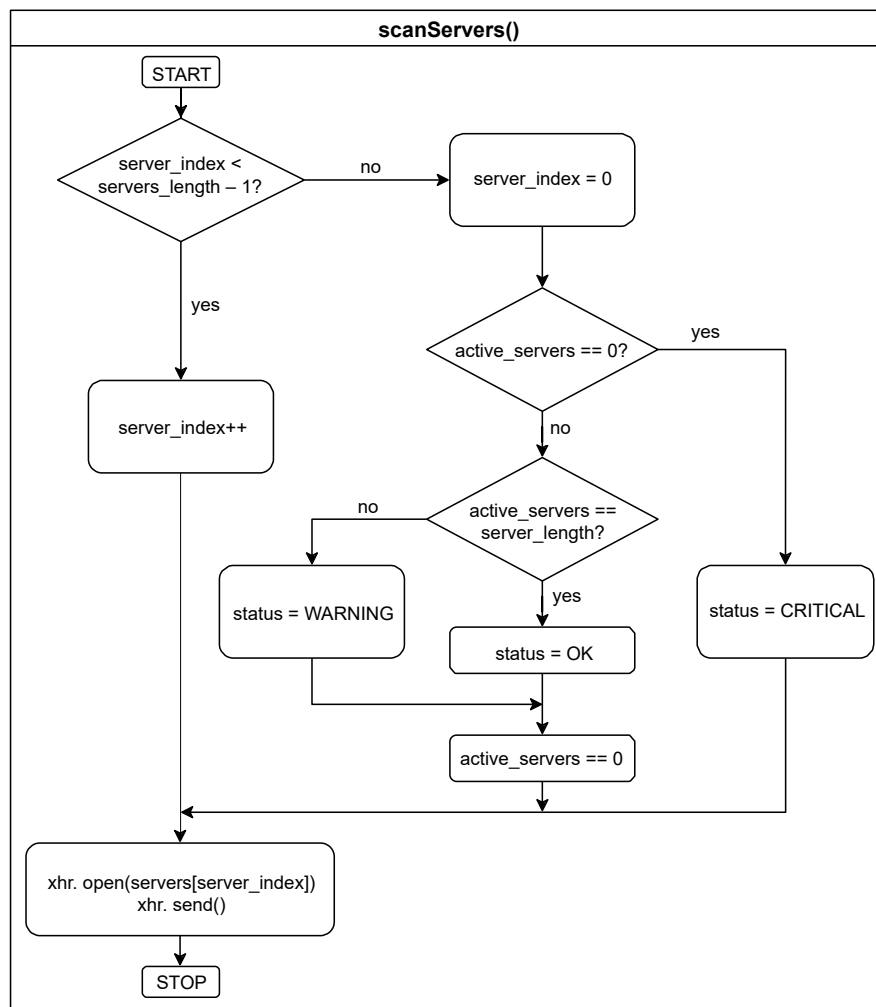


Figure 5. scanServers() procedure of the server availability tracking algorithm

errors with the server, then *svrError()* is called, which is considered a failure of the currently processed host. Subsequent calls to *scanServers()* are re-established in *svrMonitor()*.

Conclusions

The sustainable availability of web services remains crucial for their proper development. As

Reuters reported (<https://www.reuters.com/article/idUSKBN2B20NT/>), due to a fire in an OVHcloud SBG2 data center on the night of March 9–10, 2021, millions of web services went down. It also arose that some of its backups were stored in the same burned-out center, preventing users from restoring their services (<https://www.transatlantic-lawyer.com/ovh-must-pay-more-than-400000-e-after-a-fire-destroyed-itsdata-centers-why-this-decision->

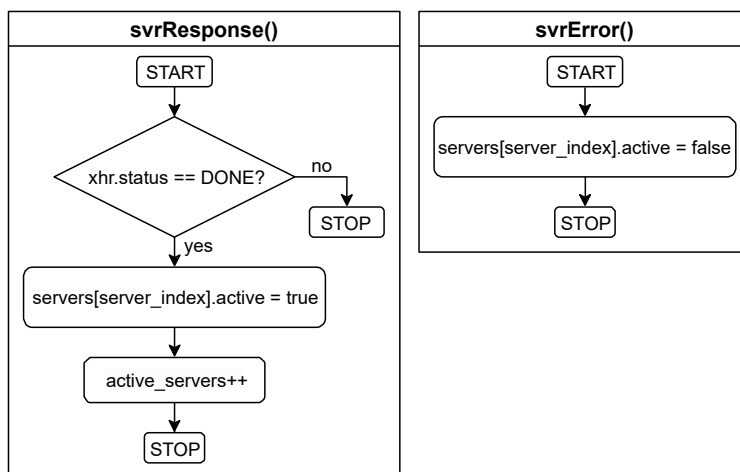


Figure 6. `svrResponse()` and `svrError()` procedures of the server availability tracking algorithm

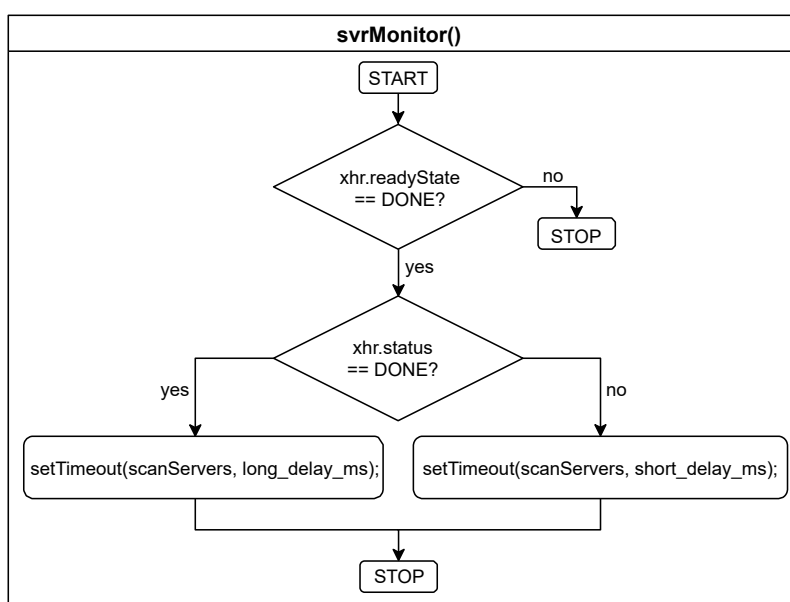


Figure 7. `svrMonitor()` procedure of the server availability tracking algorithm

is-important-for-hosting-providers-hosting-eu-personal-data/). Clearly, HA problems are vital and sometimes left unsolved by cloud providers.

In this paper, the extension of a standard single-provider HA model into the multi-cloud one is proposed. It is free from SPOFs and is scalable. It is especially applicable for internet-oriented apps when HA is required, such as supply ordering service, warehouse management, or multimedia streaming platforms. The proposed system has been implemented practically in an audio streaming service based on two web cloud providers. On the downside, it is more expensive because of the costs of several cloud suppliers. Also, the client's app requires a server availability tracking algorithm to be implemented. To fulfill demand no. 4 in the *Methods* section, the system should be designed in such a way

that it is possible to augment the struct `servers` (Table 3) using any active server. This would allow for the addition of an extra fresh host to the service in case of serious malfunction or DDoS attacks. Satisfying demand no. 5 means a careful client app design and tests that consider different malfunction scenarios.

References

1. AHMED, K.M.U., BOLLEN, M.H.J. & ALVAREZ, M. (2021) A review of data centers energy consumption and reliability modeling. *IEEE Access* 9, pp. 152536–152563, doi: 10.1109/ACCESS.2021.3125092.
2. ANSI/BICSI (2019) *Data Center Design and Implementation Best Practices*. ANSI/BICSI, 002-2019. Tampa, FL, USA: BICSI.
3. ATCHISON, L. (2020) *Architecting for Scale*. Second Edition. Sebastopol, CA, USA: O'Reilly Media, Inc.

4. CARAPOLA, A. (2018) *The Data Center Builder's Bible*. s. 1. NewVista Advisors, LLC.
5. Cisco (2024) *Cisco Validated Design Zone*. [Online]. Available at: <https://www.cisco.com/c/en/us/solutions/design-zone.html> [Accessed: January 30, 2024].
6. CLEMENT, S., BURDETT, K., RTEIL, N., WYNNE, A. & KENNY, R. (2023) Is hot IT a false economy? An analysis of server and data center energy efficiency as temperatures rise. *IEEE Transactions on Sustainable Computing*, pp. 1–12, doi: 10.1109/TSUSC.2023.3336801.
7. FERRER, A.J., PÉREZ, D.G. & GONZÁLEZ, R.S. (2016) Multi-cloud platform-as-a-service model, functionalities and approaches. *Procedia Computer Science* 97, pp. 63–72, doi: 10.1016/j.procs.2016.08.281.
8. GENG, H. (2021) *Data Center Handbook: Plan, Design, Build, and Operations of a Smart Data Center*. Palo Alto: John Wiley & Sons.
9. JADHAV, M. & CHAUDHARI, P. (2015) Energy performance optimization of server room HVAC system. *International Journal of Thermal Technologies* 5 (3), pp. 232–237.
10. LANG, M., WIESCHE, M. & KRČMAR, H. (2018) Criteria for selecting cloud service providers: A Delphi study of quality-of-service attributes. *Information & Management* 55 (6), pp. 746–758, doi: 10.1016/j.im.2018.03.004.
11. LOWE, S.D., GREEN, J. & DAVIS, D. (2016) *Building a Modern Data Center: Principles & Strategies of Design*. Bluffton, SC, USA: ActualTech Media.
12. MELL, P. & GRANCE, T. (2011) *The NIST Definition of Cloud Computing*. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available from: <https://doi.org/10.6028/NIST.SP.800-145> [Accessed: January 30, 2024].
13. PETCU, D. (2013) *Multi-Cloud: Expectations and Current Approaches*. New York, ACM Press.
14. RAMAMURTHY, A., SAURABH, S., GHAROTE, M. & LODHA, S. (2020) Selection of cloud service providers for hosting web applications in a multi-cloud environment. *IEEE International Conference on Services Computing (SCC)*, Beijing, China, pp. 202–209, doi: 10.1109/SCC49832.2020.00034.
15. RUPARELIA, N. (2015) *Cloud Computing*. Cambridge, MA: The MIT Press.
16. SEN, P., SARDDAR, D., SINHA, S.K. & PANDIT, R. (2019) Web service scheduling in multi-cloud environment. *International Journal of Computer Sciences and Engineering* 7 (1), pp. 30–38.

Cite as: Wirski, R.T. (2024) High availability model of a web service from a system administrator's perspective. *Scientific Journals of the Maritime University of Szczecin, Zeszyty Naukowe Politechniki Morskiej w Szczecinie* 77 (149), 99–106.