Towards the identification of context in 5G infrastructures

Chrysostomos Symvoulidis, Ilias Tsoumas, and Dimosthenis Kyriazis

University of Piraeus, Piraeus, Attica 18532, Greece,
{simvoul, itsoum, dimos}@unipi.gr

Abstract. The evolution of communication networks, bringing the fifth generation (5G) of mobile communications in the foreground, gives the vertical industries opportunities that were not possible until now. Flexible network and computing infrastructure management can be achieved, hence bringing more freedom to the service providers, to maximize the performance of their computing resources. Still, challenges regarding the orchestration of these resources may arise. For this reason, an engine that can recognize possible factors that affect the use of these resources and come up with solutions when needed in real-time, is required. In this paper we present a novel Complex Event Processing that is enriched with Machine Learning capabilities in order to be fully adaptive to its environment, as a solution for monitoring application components deployed in 5G infrastructures. The proposed engine utilizes Incremental DBSCAN to identify the normal behavior of the deployed services and adjust the rules accordingly.

Keywords: 5G; complex event processing, dynamic complex event processing, anomaly detection, event identification, context awareness

1 Introduction

Unlike previous generations, 5G will be more than just a mobile network. According to Soldani et al. [5] and Moyse [6] 5G will enable a reliable and efficient way of deploying several services in multiple distant facilities. The utilization of such diverse infrastructures should lead us to considerable advantages. Numerous paradigms and methodologies such as Software-as-a-Service (SaaS) and Infrastructure-as-a-Service (IaaS), etc. can be now coupled [6], thus leading to greater performance of the services per se. Services will be substantially more efficient, since the service providers will have the ability to orchestrate them, even in the cases where they are deployed in heterogeneous infrastructures. Monitoring would present a more complete image to the service providers, showing aggregated results on all their resources on all infrastructures [7], [8], [9].

As already mentioned 5G is an emerging technology that will bring many advantages both for the end-users, and the service providers, due to its increased bandwidth, high throughput and particularly high reliability [4]. But in order to take full advantage of 5G infrastructure capabilities and its functionalities, an intelligent way to orchestrate both the network and the infrastructure is necessary. Anomaly detection in such diverse environments consisting of several factors is a demanding process, hence an intelligent and interactive way to bypass such issues is required.

Complex Event Processing (CEP) is a technology, mainly used for the analysis and the process series of correlated events, using a set of techniques and tools [12]. CEP is a perfect fit for the aforementioned issues, since it is a reactive alternative to process incoming information in order to identify complex states and propose solutions to problems instantaneously.

Given the fact that in 5G infrastructures anomaly detection is a complex process, common anomaly detection solutions could not work effectively. Hence, we present a novel CEP engine for anomaly detection in such infrastructures. This engine mainly focuses on the identification of events that affect the deployment and the performance of both the applications and the network per se. The engine has been developed using the Drools Fusion framework [14], a tool created by RedHat JBoss with respect to operating CEP jobs. The proposed approach also includes an intelligent mechanism that can make the CEP engine more adaptive to the current environment, via the exploitation of Machine Learning techniques, in order to provide more customized results based on its environment every time, thus making the CEP engine context aware.

The rest of the paper is organized as follows: Chapter 2 focuses on the delivery of the state of the art in the area of network anomaly focusing mainly in the new era networks, and the CEP architectures that are currently being used. Chapter 3 depicts the overall architecture of the proposed engine. Chapter 4 regards the experimental evaluation of the engine and depicts the results. Finally, Chapter 5 concludes the paper and suggests future work that can be done based on the current engine.

2 Related Work

This section provides an overview of the current State of the Art in the CEP and Anomaly Detection areas.

2.1 Complex Event Processing

As already declared, CEP is a technique that is primarily used for the identification of certain situations (events), through the process of streams of information from various sources. Its purpose is to identify such meaningful events and respond quickly. CEP is part of the Rule Engines area [15]. CEP can be used along other similar technologies, such as Business Process Management (BPM) or Business Activity monitoring, but its distinguished due to its agility [16] in comparison with the rest. CEP is widely used in a great amount of information systems, including network monitoring and anomaly detection, business process automation (BPA) and business intelligence (BI), among others. CEP processors are able to identify events immediately, thus speeding up the decision-making process. CEP engines can identify events from numerous data streams that are often correlated, identify possible event patterns and situations, as well as alerting when necessary and trigger actions that have to be done [13]. An integral element of the CEP engines is the exploitation of potential incidents, namely the *events*, that are in general, things that happen in a certain environment, or changes of state in a current situation. Yet, the creation of events is not a simple process, on the grounds that complex events can exist through the combination of several individual events.

Another highly important term of the CEP engines regards the *Sliding Windows*. The Sliding Windows are a way of grouping (correlating) complex events together based on some factors. A widely used technique to combine such events is done through the notion of time (e.g. a number of events are occurred in a specified time window), or by the number of times they are occurred (e.g. an event X has occurred Y times).

One of the first known systems in the CEP area is Aurora [17]. Aurora supported a number of basic operations including Filtering, Resampling and Tumbling that could be applied into streams of information in a window time, or filters such as joins, maps and GroupBys that could be applied into tuples of data, once a time. Borealis [17] came as the second generation of Aurora that could operate as distributed stream processing engine. Through Borealis dynamical revision of query results and modification of the query specifications could be achieved.

Gyllstrom et al. [18] proposed SASE+, a new approach which described a temporal sequence of events that are correlated based on their attribute values. SASE+ was developed in the University of Massachusetts. Endler et al. [24] presented a semantic model for streaming data as an extension of a CEP engine in order to provide Semantic reasoning. Brenna et al. [20] in their work presented CAYUGA, a CEP engine that uses non-finite automation to match event patterns to queries, through an SQL-like language. Bhargavi et al. [19], proposed an adaptive CEP engine that can update its Knowledge Base at run-time in order to make it more dynamic and interactive. Petersen et al. [23] in their work have presented a CEP rule alteration engine using Online Learning techniques, thus making the engine more efficient over time, since the rules would adapt better to the current state of the environment.

In terms of production CEP tools, WSO2 created a CEP engine called WSO2 Complex Event Processor [10], [11]. In its core, the WSO2 CEP uses the Siddhi query engine [21] which describes each event as a tuple. It provides a lot of features including time windows, filters, joins, event patterns, event partitions and it focuses mostly on the real-time analyzation of the identified events.A robust and versatile known CEP engine is Esper [22]. Esper describes an attribute of an event as a composition of other events, it divides each event in fragments and offers domain-specific information per event. It uses an extension of SQL, the Esper Query Language (EQL), which offers filters such as sliding windows and patterns over the streams of information. Other systems that could operate CEP processes include Cordies [40], LogCEP [39], and SPA [34], along with

Apache Flink [47], [36], Apache Storm [46], [35] and Apache Spark [45], [37]; some general purpose frameworks, that could also work as CEP engines.

2.2 Anomaly Detection

Anomaly detection is used broadly in communication networks. Depending on the situation there are numerous ways to analyze and detect anomalies [26]; classification based, clustering-based, or using statistical methods and information theory. Deljac et al. in [26], [25], where they used a Bayesian network for the outage detection in telecommunication networks. Clustering-based techniques regard unsupervised learning engines, mainly for unlabeled data. Ahmed et al. in [27] used regular clustering techniques to cluster the data and detect outliers to improve the accuracy of anomaly detection engines. In [28] Ester et al. used the DBSCAN algorithm to discover noise in databases. Statistical-oriented methods are widely used in anomaly detection, including [29], that used signal processing techniques in order to identify anomalies in IP networks. Shyu et al. [30] proposed a novel paradigm to handle and analyze multidimensional network traffic datasets that could identify whether an instance in the dataset regarded an anomaly or not.

Yang et al. [31] presented a rule-based Intrusion Detection System (IDS) for Supervisory Control and Data Acquisition (SCADA) networks, using an in-depth analysis and deep packet inspection techniques.Balabine and Velednitsky in [33] published a patent that uses Support Vector Machines (SVM) for network traffic anomaly detection. Neural networks are also used for such purposes, like the one presented by Poojitha et al. in [32] where they tried to detect anomalies in a given dataset with labeled information.

Taking under consideration all the aforementioned approaches, an issue that has not been answered yet, regards the ability of a given CEP engine to be fully adaptive to the environment it concerns and provide customized insights. For this reason, our approach regards a CEP engine in order to combine these two areas and provide better results in diverse ecosystems such as a 5G infrastructure. Our approach aims at providing an engine that can be adaptive and support the deployment of applications and services in such environments trouble-free with customized events, taking under consideration the current conditions.

3 Overall architecture

This chapter describes the architecture of the developed engine. More specifically, it describes the mechanisms responsible for the collection of the data from the various components, along with the way that data are fed to the CEP engine, how this engine matches the real-time data with the rules in the Knowledge Base (KB), and finally what is our solution to the creation of an adaptive CEP engine.

The proposed engine is divided in six main steps and are depicted in Figure 1: (i) the creation of the rules that are matched to the events by the domain expert,

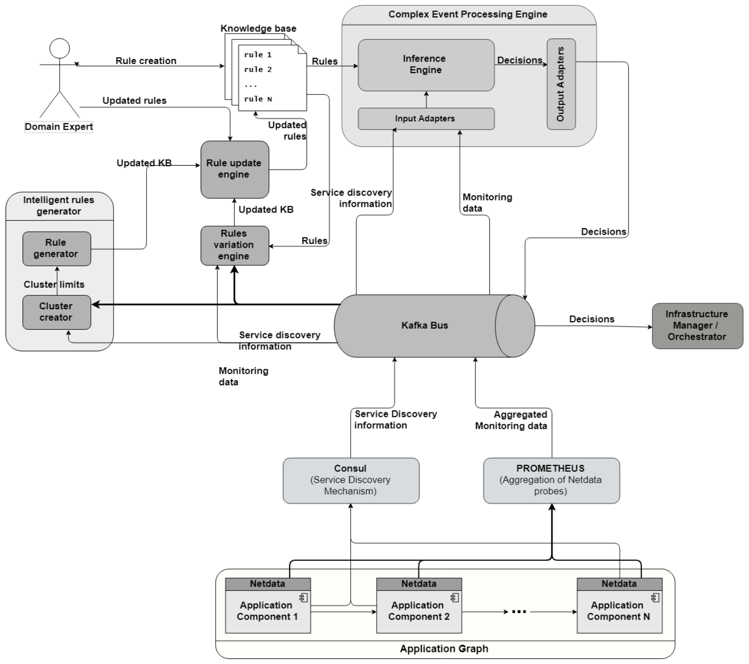


Fig. 1. CEP Engine overall architecture

(ii) the collection of the data by the various monitoring engines, (iii) the data cleaning part and transformation of the data into CEP-utilizable information, (vi) the event processing part where the streams of information are matched to some rules stored in the KB, (v) the intelligent rules generator; the engine responsible for adapting to the environment and update the rules accordingly, and finally (vi) the propagation of the decisions made by the CEP engine to the interested mechanisms. These steps are described in detail in the sections below.

3.1 Creation of the events

A domain expert of 5G networks and infrastructures is asked to identify events that affect the performance of the network and the resource infrastructures. The events are mostly related to the resource usage of the computing resources and the network usage. These events are then "translated" into rules and are used for the identification of the events by the engine. This translation regards the creation of rules based on the decisions of the domain expert.

3.2 Data collection

The process continues with the collection of the data by the various monitoring engines. The architecture suggests using Netdata [41] probes in each application component, that collect information related to the usage of the machine, including CPU, Memory and Disk usage percentage, currently running applications, system up-time, IPv4 / IPv6 received, sent and aborted packets and Kbps, etc. These application components regard Virtual Machines, Containers, or even Hardware computing resources (desktop PCs, etc.).

As depicted in Figure 1, the collected data are scraped by the Prometheus monitoring engine [42]. Its job is to provide aggregated results with respect to the whole infrastructure and specific results per deployed application component, in order for both the CEP engine and the Intelligent rules generator to have an overall image of the current state of the facility.

Additionally, data are collected by the Consul [43] service discovery mechanism too. Its purpose is to collect valuable information related to the services, including health checking, service communication, and to check whether the APIs of the services are functional, etc.

All the data that are collected by Prometheus and Consul are then sent to a Kafka Bus [44], as the proposed Publish / Subscribe (PubSub) platform. From there anyone that is interested in retrieving that data, can subscribe to each topic and collect them.

3.3 Data cleaning

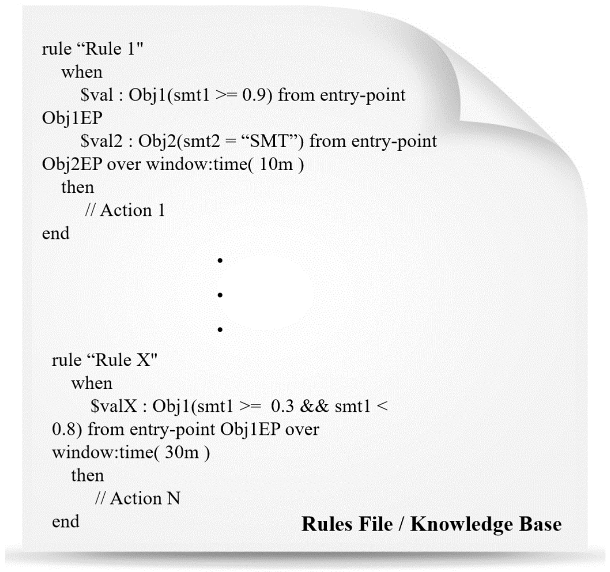
As soon as the data are collected from the Prometheus engine and the Consul mechanism from the input adapters of the CEP engine, the data cleaning process is instantiated. This process regards mainly the data collected from the Prometheus system. Netdata and Prometheus collect a great number of information that only a just a small amount of data are used by the CEP engine. In addition, the data are received in a form that the CEP engine cannot handle, so they are transformed into CEP engine - utilizable information, and are finally fed to the engine.

3.4 Complex Event Processing

After the collection and data cleaning process is complete, the CEP engine is now able to initiate the event identification process. Drools Fusion [14] is the preferred CEP engine. The reason that Drools Fusion was selected is the fact that is has several advantages on the way the events are handled, its ease of creation rules and has numerous useful features, including Sliding Time and Length Windows, support of both Stream and Batch Processing and several ways of grouping events together apart from Sliding Windows.

Before the collection of the data, the engine collects the rules from the KB. These rules are the representation of the events and are in the form of WHEN ... THEN ..., as also presented in Figure 2.

After the streaming data are collected and cleaned they are forwarded to the CEP engine. Here starts the process of the identification of complex events. The engine collects the rules from the KB and when the incoming data match a rule, that specific rule gets triggered.



```
rule "Rule 1"
   when
       $val : Obj1(smt1 >= 0.9) from entry-point
Obj1EP
       $val2 : Obj2(smt2 = "SMT") from entry-point
Obj2EP over window:time( 10m )
   then
       // Action 1
end
                     •
                     •
                     •
 rule "Rule X"
    when
        $valX : Obj1(smt1 >=  0.3 && smt1 <
 0.8) from entry-point Obj1EP over
 window:time( 30m )
    then
        // Action N
 end
```

Rules File / Knowledge Base

Fig. 2. Rules file example

3.5 Personalized events on the fly

This is the most important part of the proposed engine. The proposed engine includes a *dual mode of operations* regarding the update of the KB.

The first mode regards the *Rules variation engine*. Its purpose is to check the numerical difference between the streaming data and the expressions in the rules (i.e the expression in the WHEN part, Figure 2, that triggers the rule). If that difference is greater that a given amount, the rules are updated in order to reduce that void, in cases where it is considered necessary. Also, through this engine, the domain expert can update the KB personally, by modifying, adding, or deleting rules. Then a listener notifies the inference engine, in order to act based on the updated KB.

The second mode, regards a more sophisticated way of updating the KB in a more personalized manner, with respect to the deployed application graphs, using Machine Learning techniques. The intention of this module, is to identify the normal behavior of the deployed applications and services, based on the data streams provided by the monitoring and the service discovery mechanisms, in order to create customized rules. In such diverse environments such as a 5G infrastructure, even the domain experts cannot be fully aware of the current state of the infrastructure, so the events are more generalized. Hence, the existence of a mechanism that can adapt on the current state of the environment is necessary.

Therefore, we developed an engine that uses *Incremental Density Based Spatial Clustering of Applications with Noise (Incremental DBSCAN)* [38], an enhancement of the known DBSCAN algorithm that can work with continuously incoming data.

Using DBSCAN the engine can identify the normal behavior of the application graph and the corresponding application components and adjust the rules accordingly, and group it into clusters. In more detail, the *Intelligent rules generator* component, as depicted in Figure 1, is comprised of two major components; the *Cluster creator* and the *Rule generator*.

The Cluster creator collects the data from the Kafka bus and with the use of Incremental DBSCAN, creates clusters that present the behavior of the networking and computing resources of the application components. It then discovers the limits of the major cluster that regards the normal behavior of the component. The cluster that is considered to be the one that depicts the normal behavior is the one with the most elements in it.

The limits of the clusters are then forwarded to the Rule Generator. Its job is to update the rules in the KB, now based on the outcomes of the Cluster creator. After the update of the KB is complete, the listener notifies the CEP engine.

3.6 Propagation of the decisions made by the CEP engine

After the rules are triggered, the process of communicating the decisions begins. The outcome of every rule that is triggered is sent to the Kafka Bus that is described above and the corresponding services and engines have to be subscribed to the topic related to the CEP, in order to get notified.

4 Experimental Evaluation

This chapter presents the experiments that the developed CEP is put under. Due to the absence of a real 5G infrastructure, an experiment was executed in a comparable environment.

4.1 Working environments

For the testing of the CEP engine we used two Virtual Machines (VM) that are comprised of 2 vCPUs, 8GB of memory and 20GB of Hard Disk. They both ran the Ubuntu 14.04 operating system. During the test we put these VMs under pressure using benchmarks. Another VM with the same computing resources had the responsibility to collect the monitoring data and the service discovery information from Prometheus and Consul and forward them to the Kafka bus.

The CEP engine, as well as the rule generator engine ran on a desktop PC with Windows 10 operating system, with Intel dual-core i3 CPU, 8GB of RAM and 500GB of Hard Disk Drive.

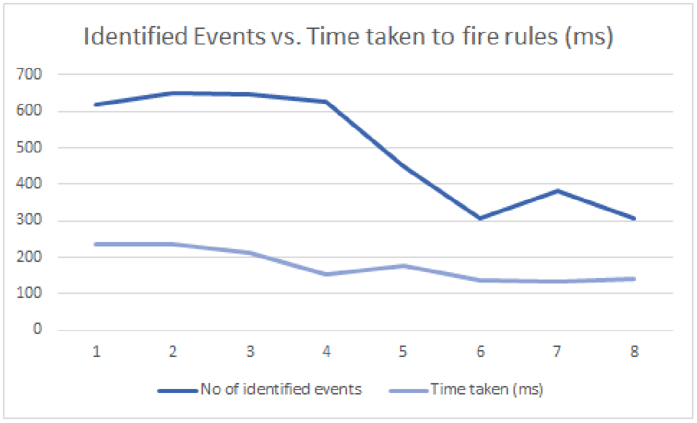


Fig. 3. Number of Identified events vs. Time taken to fire rules (ms) in a 8-hour period

4.2 Results

For the evaluation part, we are based on metrics that can describe the overall performance of the engine. More specifically, these metrics regard the number of the rules that are fired, the total time taken to fire the corresponding rules from the time the data arrived in the engine, and if that changed when the amount of the rules triggered was increased.

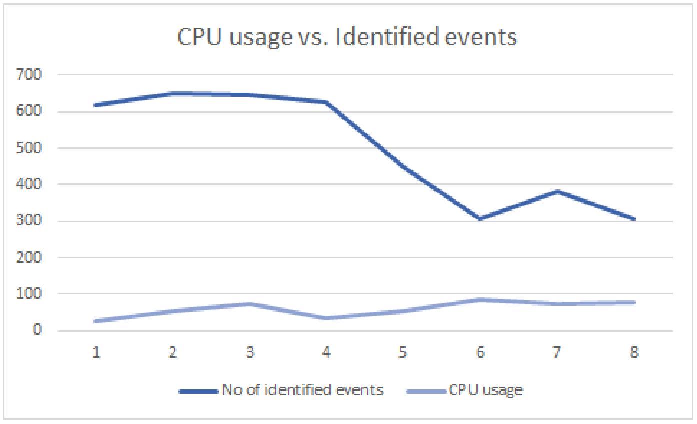


Fig. 4. CPU usage vs. Identified events in a 8-hour period

The CEP engine presented indeed the current situation of the resources with minimum delay, since the events were being identified in less than 300ms from the moment the data were received in the engine, regardless of the number of the number of the rules that were being triggered at the same time, as also depicted in Figure 3.

In Figure 4 the number of the rules that were being triggered is depicted, in comparison to the CPU usage. At first the number of the identified events was quite large. This was due to the fact that the rules that were used at first were the predefined ones. After the Rule generator engine was activated, the number of the identified events dropped significantly. Still though, events that were essential and described a change of state in the environment were identified.

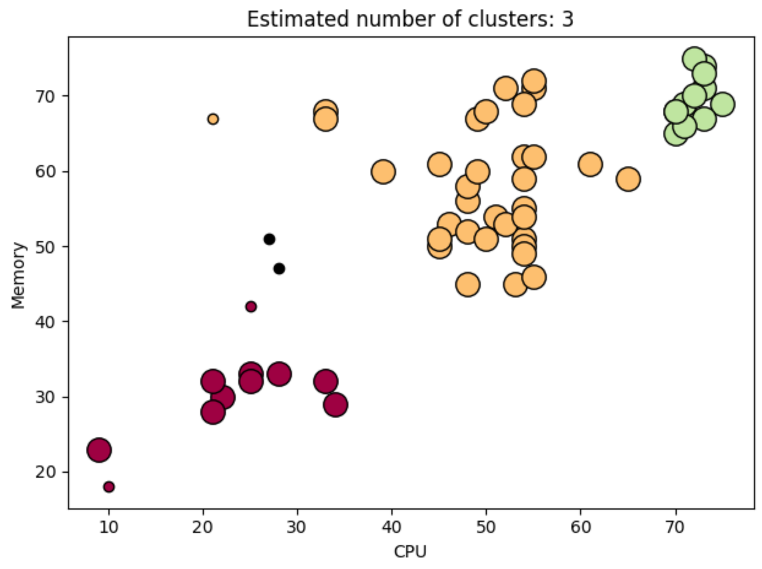


Fig. 5. Clusters created by the Intelligent rules generator engine

Figure 5 presents the clusters that were created by the DBSCAN algorithm. The cluster with the orange color was identified as the normal behavior of the deployed VMs. The limits of the cluster (i.e minimum CPU, Memory, Disk usage etc.) were forwarded to the Rule generator, in order to update the KB. When the process was complete the CEP engine identified as an event anything outside these boundaries. This led to the creation of events that were not general, but were adapted to the specific situation.

5 Conclusion and Future Work

The identification of factors that affect the use of such diverse infrastructures and networks, such as 5G is indeed a difficult and complex process. In this paper, we have presented a novel CEP engine that can overcome issues, when dealing with multidimensional data and be considered a trustworthy alternative for a more interactive network anomaly detection. The use of the Incremental DBSCAN algorithm makes the current engine more adaptive and the events identified are customized to the current situation.

It is within our future plans to extend the present approach, by making the Incremental DBSCAN more precise and continue our research in ML and CEP in order to make the engine more generic and flexible to changes in its environment, thus being capable of providing insights to other areas of interest including Cloud computing.

6 Acknowledgment

This work has received funding from the European Unions Horizon 2020 research and innovation program under grant agreement No 761898 project MATILDA.

References

1. Andrews, J. G., Buzzi, S., Choi, W., Hanly, S. V., Lozano, A., Soong, A. C., & Zhang, J. C. (2014). What will 5G be?. IEEE Journal on selected areas in communications, 32(6), 1065-1082.
2. Bangerter, B., Talwar, S., Arefi, R., & Stewart, K. (2014). Networks and devices for the 5G era. IEEE Communications Magazine, 52(2), 90-96.
3. Compton, J. (2018, September 21). 5G: Intelligent Transformative Networks, Not Just Faster Phones. Retrieved from https://www.forbes.com/sites/intelai/2018/09/21/5g-intelligent-transformative-networks-not-just-faster-phones/
4. Suriano, D. (2018, September 23). The Future Of Networking Is 5G: Businesses Must Prepare Now. Retrieved from https://www.forbes.com/sites/oracle/2018/09/24/the-future-of-networking-is-5g-businesses-must-prepare-now
5. Soldani, D., & Manzalini, A. (2014). A 5G Infrastructure for" Anything-as-a-Service". Journal of Telecommunications System & Management, 3(2), 1.
6. Moyse, I. (n.d.). 5 Reasons for a Multi-Cloud Infrastructure — Dyn Blog. Retrieved from https://dyn.com/blog/5-reasons-for-a-multi-cloud-infrastructure/
7. Uittenbogaard, T. (2015). What are the advantages of a Multi-site/Multi-domain solution and who's to benefit from it? Retrieved from https://oneshoe.com/news/what-are-advantages-multi-sitemulti-domain-solution-and-whos-benefit-it
8. Hume, A. C., Al-Hazmi, Y., Belter, B., Campowsky, K., Carril, L. M., Carrozzo, G., Engen, V., Garcia-Perez, D., Ponsati, J. J., Kubert, R., Rohr, C., van Seghbroeck, G., & Liang, Y. (2012, June). Bonfire: A multi-cloud test facility for internet of services experimentation. In International Conference on Testbeds and Research Infrastructures (pp. 81-96). Springer, Berlin, Heidelberg.
9. Baldin, I., Chase, J., Xin, Y., Mandal, A., Ruth, P., Castillo, C., Orlikowski, V., Heermann, C., & Mills, J. (2016). Exogeni: A multi-domain infrastructure-as-a-service testbed. In The GENI Book (pp. 279-315). Springer, Cham.
10. Perera, S., Sriskandarajah, S., Vivekanandalingam, M., Fremantle, P., & Weerawarana, S. (2014, May). Solving the grand challenge using an opensource CEP engine. In Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems (pp. 288-293). ACM.

11. WSO2 Complex Event Processor. Retrieved from https://wso2.com/products/complex-event-processor
12. Luckham, D. (2002). The power of events (Vol. 204). Reading: Addison-Wesley.
13. Garcia, J. (2012). A Complex Event Processing system for monitoring manufacturing systems. Tampere University of Technology. Retrieved from https://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/20958/garcia_izaguirre.pdf
14. Drools Fusion 6.2.0 documentation. Retrieved from https://docs.jboss.org/drools/release/6.2.0.CR3/drools-docs/html/
15. Fulop, L. J., Toth, G., Rcz, R., Panczel, J., Gergely, T., Beszedes, A., & Farkas, L. (2010, July). Survey on complex event processing and predictive analytics. In Proceedings of the Fifth Balkan Conference in Informatics (pp. 26-31).
16. Saboor, M., & Rengasamy, R. (2013). Designing and developing Complex Event Processing Applications. Sapient Global Markets.
17. Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., & Zdonik, S. (2002, August). Monitoring streams: a new class of data management applications. In Proceedings of the 28th international conference on Very Large Data Bases (pp. 215-226). VLDB Endowment.
18. Diao, Y., Immerman, N., & Gyllstrom, D. (2007). Sase+: An agile language for kleene closure over event streams. UMass Technical Report.
19. Bhargavi, R., Pathak, R., & Vaidehi, V. (2013, July). Dynamic complex event processingAdaptive rule engine. In Recent Trends in Information Technology (ICRTIT), 2013 International Conference on (pp. 189-194). IEEE.
20. Brenna, L., Gehrke, J., Hong, M., & Johansen, D. (2009, July). Distributed event stream processing with non-deterministic finite automata. In Proceedings of the Third ACM International Conference on Distributed Event-Based Systems (p. 3). ACM.
21. Suhothayan, S., Gajasinghe, K., Loku Narangoda, I., Chaturanga, S., Perera, S., & Nanayakkara, V. (2011, November). Siddhi: A second look at complex event processing architectures. In Proceedings of the 2011 ACM workshop on Gateway computing environments (pp. 43-50). ACM.
22. ETES Intelligence - Esper and NEsper (2006). Where Complex Event Processing meets Open Source. Esper & NEsper, 2, 2006-2013.
23. Petersen, E., To, M. A., & Maag, S. (2016, November). An online learning based approach for CEP rule generation. In Communications (LATINCOM), 2016 8th IEEE Latin-American Conference on (pp. 1-6). IEEE.
24. Endler, M., Briot, J. P., e Silva, F. S., de Almeida, V. P., & Haeusler, E. H. (2017, September). Towards stream-based reasoning and machine learning for IoT applications. In Intelligent Systems Conference (IntelliSys), 2017 (pp. 202-209). IEEE.
25. Deljac, Z., Randic, M., & Krcelic, G. (2015). Early detection of network element outages based on customer trouble calls. Decision Support Systems, 73, 57-73.
26. Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.
27. Ahmed, M., & Mahmood, A. N. (2013, June). A novel approach for outlier detection and clustering improvement. In Industrial electronics and applications (ICIEA), 2013 8th IEEE conference on (pp. 577-582). IEEE.
28. Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd (Vol. 96, No. 34, pp. 226-231).
29. Thottan, M., & Ji, C. (2003). Anomaly detection in IP networks. IEEE Transactions on signal processing, 51(8), 2191-2204.

30. Shyu, M. L., Chen, S. C., Sarinnapakorn, K., & Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier. MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING.
31. Yang, Y., McLaughlin, K., Littler, T., Sezer, S., & Wang, H. F. (2013). Rule-based intrusion detection system for SCADA networks.
32. Poojitha, G., Kumar, K. N., & Reddy, P. J. (2010, July). Intrusion detection using artificial neural network. In Computing Communication and Networking Technologies (ICCCNT), 2010 International Conference on (pp. 1-7). IEEE.
33. Balabine, I., & Velednitsky, A. (2017). U.S. Patent No. 9,843,488. Washington, DC: U.S. Patent and Trademark Office.
34. Dijkman, R., Peters, S., & ter Hofstede, A. (2016, September). A toolkit for streaming process data analysis. In Enterprise Distributed Object Computing Workshop (EDOCW), 2016 IEEE 20th International (pp. 1-9). IEEE.
35. Gaunitz, B., Roth, M., & Franczyk, B. (2015, April). Dynamic and scalable real-time analytics in logistics combining Apache Storm with Complex Event Processing for enabling new business models in logistics. In Evaluation of Novel Approaches to Software Engineering (ENASE), 2015 International Conference on (pp. 289-294). IEEE.
36. Bansod, R., Kadarkar, S., Virk, R., Raval, M., Rashinkar, R., & Nambiar, M. (2018, June). High Performance Distributed In-Memory Architectures for Trade Surveillance System. In 2018 17th International Symposium on Parallel and Distributed Computing (ISPDC) (pp. 101-108). IEEE.
37. Liu, G., Zhu, W., Saunders, C., Gao, F., & Yu, Y. (2015). Real-time complex event processing and analytics for smart grid. Procedia Computer Science, 61, 113-119.
38. Chakraborty, S., & Nagwani, N. K. (2014). Analysis and study of Incremental DBSCAN clustering algorithm. arXiv preprint arXiv:1406.4754.
39. Cao, J., Wei, X., Liu, Y. Q., Mao, D., & Cai, Q. (2014). LogCEP-Complex event processing based on pushdown automaton. International Journal of Hybrid Information Technology, 7(6), 71-82.
40. Koch, G. G., Koldehofe, B., & Rothermel, K. (2010, July). Cordies: expressive event correlation in distributed systems. In Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems (pp. 26-37). ACM.
41. NetData. Retrieved from https://my-netdata.io/
42. Prometheus monitoring system. Retrieved from https://prometheus.io/
43. Consul by HashiCorp. Retrieved from https://consul.io/
44. Apache Kafka. Retrieved from https://apache.kafka.org/
45. Apache Spark: Lightning-fast cluster computing. Retrieved from http://spark. apache. org
46. Marz, N. (2013). Storm: Distributed and fault-tolerant realtime computation. 2011-10-21]. Retrieved from https://www. infoq, com presentations/Storm-Introduction.
47. Flink, A. (2016). Scalable batch and stream data processing. Retrieved from https://flink. apache. org.