J Ambient Intell Human Comput (2018) 9:1097–1117
DOI 10.1007/s12652-017-0542-0

ORIGINAL RESEARCH

Orchestration of use-case driven analytics in 5G scenarios

Lorena Isabel Barona López[1] · Jorge Maestre Vidal[1] · Luis Javier García Villalba[1]

Received: 13 February 2017 / Accepted: 30 June 2017 / Published online: 17 July 2017

Abstract The SELFNET project provides an autonomic network management framework for 5G networks with a high degree of automation, self-healing and self-optimization. These capabilities are achieved through a layered architecture and a use-case driven approach. A differentiating feature on SELFNET is its competence when creating and customizing new use cases and their related virtual functions. In this way, the use case operators are able to introduce new rules and parameters that will be taken into account in the analysis and decision-making tasks. Due these characteristics, the orchestration of its analytical functions poses an important challenge in terms of configurability, synchronization and management of resources. In order to contribute to their resolution, this paper aims to lay the groundwork for implement the design and specification of the SELFNET Analyzer orchestration. To this end, several key issues related with the internal coordination of the analytics are introduced, among them initial assumptions, design principles, limitations, partitioning of the analysis process, data persistency and optimization. The proposed orchestration strategy has been implemented with different uses cases within the SELFNET Project.

✉ Luis Javier García Villalba
javiergv@fdi.ucm.es

Lorena Isabel Barona López
lorebaro@ucm.es

Jorge Maestre Vidal
jmaestre@ucm.es

[1] Group of Analysis, Security and Systems (GASS), Department of Software Engineering and Artificial Intelligence (DISIA), Faculty of Computer Science and Engineering, Office 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases, 9, Ciudad Universitaria, 28040 Madrid, Spain

Keywords 5G · Situational awareness · SDN/NFV · Data analysis · Orchestration

1 Introduction

The amount and complexity of cyber threats have risen alarmingly in recent years (ENISA 2015). Because of this, the information security management plays a very important role in the strategies of large organizations. Several guidelines and platforms for its implementation have been published [ISO/IEC 27000 (ISO 2005), NIST-SP 800 (NIST 2007), CVSS-SIG-First (CVSS 2015), etc.], but despite its effectiveness in conventional scenarios, it has been shown that they do not adequately operate in dynamic monitoring environments (Webb et al. 2014). This is the case of complex use cases, where the circumstances in which observations are made directly affect the ability of decision-making. In this context, examples of common issues when identifying the best mitigation/optimization actions are: inadequate asset assessment, fluctuations at data sources, difficulties when configuring new uses cases, and lack of scalability or interoperability.

In order to tackle these problems, there is a tendency to assume more cognitive methodologies, thereby facilitating understanding the environment through contextual analysis. High among those is the development of the Situational Awareness (SA) of the protected environment by applying the Endsley's model (Endsley 1988). In accordance with this method, the perception, comprehension and projection of the system status must be taken into account. As defined by Endsley, the term situational awareness refers to "the perception of the elements in the environment within a volume of time and space, comprehension of their meaning and the projection of their status in the near future", implicitly stressing how important the context is. As a result of the enormous complexity that entails managing the security of current networks, the Endsley's model has been specifically adapted to these scenarios, which has led to coining the term Network Security Situational Awareness (NSSA) (Leau et al. 2015).

Bearing this in mind, 5G networks, as clear examples of complex and dynamic monitoring environments are the focus of the research proposed in this paper. These technologies try to meet the requirements that are expected to be demanded by the current communication schemes in the short and long terms. As stated by Osseiran et al. (2014), they may be summarized in three great challenges: (1) enhancement of latency and reliability by supporting use-case dependent capabilities, such as the deployment of specific purpose applications, among them health-care, logistics, security or incidence response tools; (2) 5G must support a wide range of data rates with very high availability and reliability; (3) finally, in order to facilitate the inclusion of a large number of devices, networks must be scalable and flexible. Note that these endpoints must be simple enough to do not pose high battery consumption. In general terms, advances towards 5G technologies are based on combining and integrating a large number of emerging technologies, such as Network Function Virtualization (NFV) (Mijumbi et al. 2016), Software Defined Networking (SDN) (Xia et al. 2015), Device to Device Communications (D2D) (Qiao et al. 2015); and analytic tools for network awareness, among them Artificial Intelligence (AI), Big Data or Self-Organized Networks (SON) (Baldo et al. 2014).

At present, there are different projects aimed at facilitating the integration of these technologies into 5G scenarios. Significant efforts have been done by the European Commission under 5G-PPP and Horizon H2020 programs in order to support the new generation of mobile networks. It has led to the foundation of the 5G-PPP partnerships, which is committed to foster 5G advances in different strands such as cognitive network management or 5G Network Security (5G-PPP 2017). Table 1 summarizes some of the projects involved in this association. Their differences and similarities are discussed in depth by Barona López et al. (2016). Notable among them is the SELFNET approach (SELFNET 2014), where an autonomic management framework to provide network intelligence and self-organizing capability for 5G mobile network infrastructures is provided.

SELFNET includes the widest variety of cutting-edge technologies and adapts the Endsley's model (Endsley 1988), as well as the NSSA paradigm, to the 5G scene, as it is described by Barona López et al. (2017a, b). The latest effort toward providing SELFNET of an analytical

Table 1 Research projects on mobile networks

Project	Related technologies	Use cases
MCN (2013)	SDN, Cloud Computing	(1) Cloud Computing for mobile network operations, (2) end-to-end mobile Cloud
T-NOVA (2013)	SDN, NFV	High-level scenario, (2) VNFs, (3) service chaining
UNIFY (2013)	SDN, NFV	(1) Infrastructure virtualization, (2) flexible service chaining, (3) network service chain invocation for providers
CROWD (2013)	SDN, SON	General purpose
5G-NORMA (2014)	SDN, NFV	(1) Multi-service, (2) multi-tenancy
CHARISMA (2014)	SDN, NFV	General purpose
SELFNET (2014)	SDN, NFV, SON, Cloud Computing	(1) Self-healing, (2) self-optimization, (3) self-protection
COGNET (Xu et al. 2016)	SDN, NFV, machine learning	(1) Situational context, (2) just-in-time services, (3) user-centric services, (4) optimized services, (5) SLA enforcement, (6) collaborative resource management
5G-Ensure Project (2014)	SDN, NFV, security models	11 Use case clusters: (1–4) identities, authentication, authorization and privacy, (5) software-defined networks, virtualization and monitoring, (6–10) availability, reliability and integrity and (11) lawful interception
SONATA (2014)	SDN, NFV, cloud	(1) Internet of things, (2) virtual CDN, (3) guaranteed, resilient and secure service delivery in industrial networks, (4) vEPC, (5) personal security applications, (6) client and hosting service providers
5G-NOW (2013)	MTC, CoMP, M2M	(1) PRACH scenario, (2) GFDM, (3) uplink CoMP with joint reception, (4) multiuser uplink on fragmented spectrum with FBMC, (5) downlink CoMP with FBMC
METIS (METIS-II 2014)	SDN, Multi-RAT, D2D, M2M	Five scenarios: (1) amazingly fast, (2) great service in a crowd, (3) ubiquitous things communicating, (4) best experience follows you, (5) super real-time and reliable connections

component capable of meeting the 5G requirements in a use-case driven approach is summarized by Barona López et al. (2017b), where the design principles, architecture and the formalization of how new use cases must be onboarded are detailed. But it does not explicitly indicate how all this information is organized, as well as how the analytical process is performed. The sophistication of these tasks results on the need of develop a novel orchestration of analytics (SELFNET Analyzer Orchestrator), adapted to the 5G monitoring environment and the use-case driven politics derived from the SELFNET project, which is the main contribution of this paper. Other contributions are the specification and implementation of the dataflows in the SELFNET Analyzer Framework, the proposal of strategies for their execution and optimization, and a battery of comprehensive examples which facilitates understanding the approach, and serves as a guide for the design and deployment of similar components at future projects. This paper is organized into eight sections, being the first of them the present introduction; Sect. 2 describes the essential elements of the SELFNET project, where the Analyzer and the specification of the onboarded data is emphasized; Sect. 3 introduces the initial assumption of the SELFNET orchestrator; Sect. 4 explains its design principles; Sect. 5 details its workflow; Sect. 6 discusses the execution and optimization strategies; Sect. 7 illustrates a battery of practical examples; Finally, Sect. 8 concludes this work.

2 Background

This section describes in detail the key points of SELFNET necessary for understanding the Analyzer and its orchestration. In particular, the SELFNET architecture and its adaptation to the NSSA, the design of its Analyzer Module and the descriptors of the use cases are reviewed.

2.1 SELFNET and the situational awareness on 5G scenarios

SELFNET H2020 Project provides a smart autonomic network management framework for 5G mobile networks based on the combination of 5G key-enabled technologies: SDN, SON, NFV, Artificial Intelligence and cloud computing. SELFNET enables the autonomic deployment of virtual network functions and the reconfiguration of network parameters in order to mitigate existing or potential problems, while maintaining the Quality of Experience (QoE) of end users (Selfnet 2014). These capabilities are provided by means a layered architecture and a use-case driven approach. On the one hand, three use cases were defined: (1) self-protection capabilities to mitigate or prevent security problems such as a cyber-attack, (2) self-healing capabilities to prevent or correct network failures and (3) self-optimization to dynamically improve the service and network performance. For this purpose, SELFNET proposes two kind of advanced network functions: (1) sensors to monitor specific network information and (2) actuators to perform countermeasures to fix or mitigate possible problems. On the other hand, SELFNET architecture is based on six layers (Fig. 1): Infrastructure Layer, Data Network Layer, SON Control Layer, SON Autonomic Layer, NFV Orchestration and Management Layer and SON Access Layer, as is described by Neves et al. (2016).

- *Infrastructure Layer* It provides the physical resources required for the instantiation of virtual functions. The Physical Sublayer, Virtualization Sublayer and Cloud Computing Sublayer enable the virtualization of compute, network and storage resources.
- *Data Network Layer* The network functions (NFs) are instantiated and interconnected in a designed topology. It includes the NF required for normal operation and SON functionalities.
- *Control Layer* It includes the SON sensors and actuators. The SON sensors collect data from different sources and the SON actuators execute response actions into the network. These elements are controlled by the SON Autonomic Layer (intelligence).
- *SON Autonomic Layer* This layer is responsible for providing the network intelligence. For this purpose, the system monitors and analyse the incoming information in order to diagnosis network problems. Then, it uses the available network functions to decide the best reaction strategy. Taken decisions are sent to NFV orchestration and Management Layer.
- *NFV Orchestration and Management Layer* It controls the deployment and instantiation of the different NFs in the infrastructure. This layer follows the ETSI MANO recommendations.
- *SON Access Layer* It provides the interface used by external actors like Business Support Systems (BSS) or Operational Support Systems (OSS). Similarly, the network administrator also can stop, verify and enforce actions on SELFNET.

In turn, SON Autonomic Layer is responsible to provide the network intelligence by means Monitor and Analyzer sublayer and Autonomic Management sublayer. In particular, the Situational Awareness of SELFNET is achieved through the application of Endsley model (Endsley 1988), which define three main phases: Perception (Monitor), Comprehension (Aggregation) and Projection (Analysis and Diagnosis) as is shown in Fig. 2.

This approach supposes a high challenge because the information is gathered from different sources (monitoring

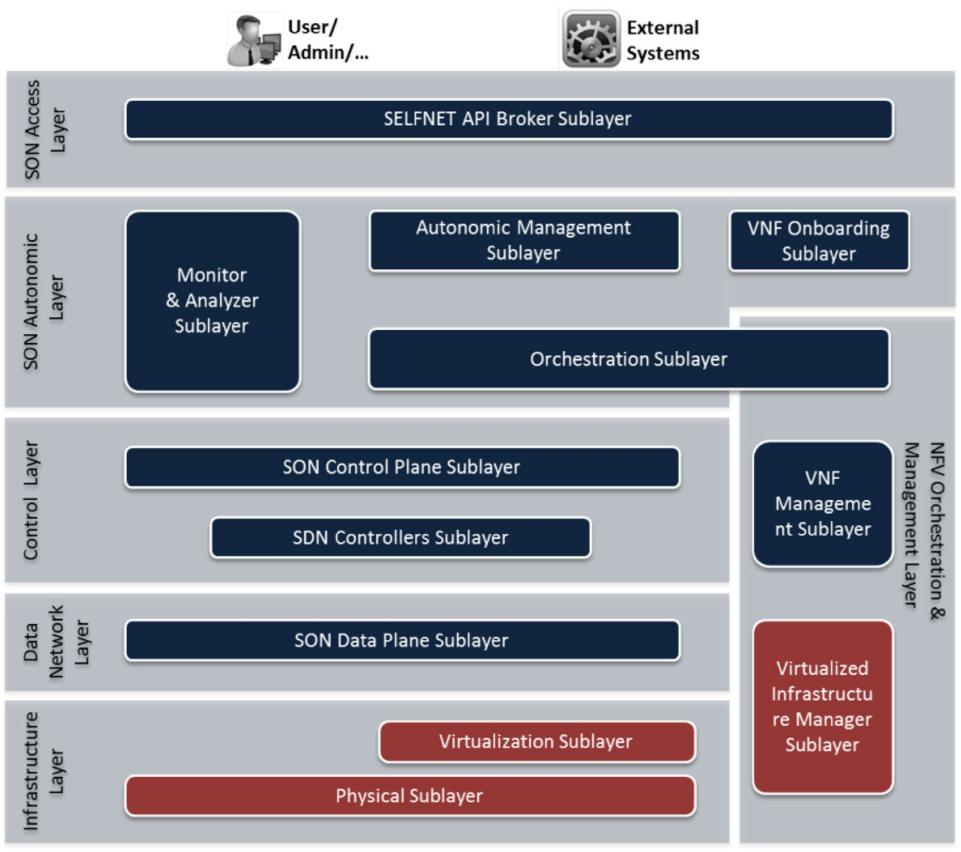


Fig. 1 SELFNET architecture

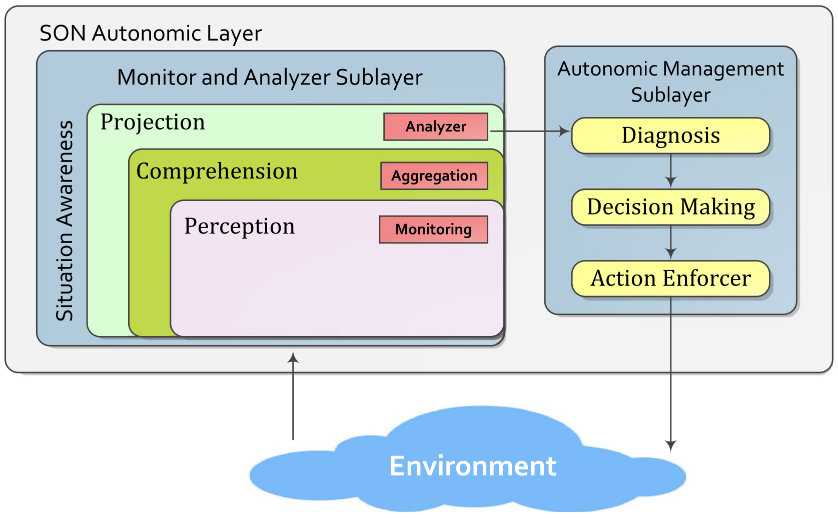


Fig. 2 Situational awareness in SELFNET project

task) and then the raw data is aggregated and correlated in order to provide high level metrics (aggregation and correlation task). In the next step, suspicious conditions are inferred or detected (analyzer task) and then they are sent to Diagnosis sublayer. Finally, this sublayer applies advanced intelligent techniques to perform proactive and reactive actions.

2.2 SELFNET analyzer: design principles and architecture

The general assumptions, requirements and the first items to consider related with the design principles of the Analyzer were previously introduced by Barona López et al. (2017b). In accordance with this publication, it must be (1) scalable, extensible and multi-level by design; (2) use-case driven, where the use-case operators are able to specify the inclusion/modification of its functionality; (3) the use case knowledge-bases required for the analytics are provided by skilled operators or by accurate machine learning algorithms; (4) user-friendly in terms of use case declaration and composition of knowledge inference rules; (5) the management of knowledge considers uncertainty and stochastic events; (6) the data sources do the filtering of the input data, hence removing inconsistencies, ambiguity and repetition on the crisp data (i.e. the SELFNET Analyzer does not perform filtering actions). All these assumptions and limitations are inherited by the orchestrator, and therefore they are considered in this proposal.

The SELFNET Analyzer relationship with the rest of the project components is summarized in Fig. 3, where a view of their main data sources is illustrated as a black box model. There two main information sources as facts Fa, were identified: Aggregation [Events $Fa(Ev)$, Thresholds $Fa(T_H)$ and Key Performance Indicators $Fa(KPI)$] and internal analytic elements [pattern recognition $Fa(PR)$, forecasts $Fa(Ft)$ and adaptive thresholds $Fa(AT_h)$]. The final conclusions that compose the SELFNET Situational Awareness are sent to Diagnosis module labeled as symptoms, via reports.

The SELFNET Analyzer architecture is shown in Fig. 4. It is centralized and their components are divided into eight main elements: Pattern Recognition (no. 1), Prediction (no. 2), Adaptive Thresholding (no. 3), Knowledge-base (no. 4), Inference Engine (no. 5), Memory (no. 6), User Interface (no. 7) and Uncertainty Estimation (no. 8). Where Pattern Recognition infers new facts related with patterns and regularities found in the aggregated data, Prediction discovers facts related with forecasting aggregated data or previously known facts, and Adaptive Thresholding establishes the limitations to be taken into account when inferring new knowledge. The core of the SELFNET Analyzer is a rule base engine composed by the Knowledge-base, Inference Engine and Memory. It applies use-case driven rules for deducting conclusions from the previously identified facts. If some of them match with situations of interest for the Diagnosis module, they are adapted by the Uncertainty Estimation component, which allows them to be interpreted as symptoms by the SELFNET upper layers. Note that the configuration of the use cases is performed at the User Interface.

2.3 Specification of the use cases

When initiated, the SELFNET Analyzer is a *tabula rasa* without actions nor reasoning to be orchestrated. It requires the onboard of use cases, which provides the script with the activities that may be performed. If a new use case is onboarded, the information that it is able to manage, as well as the analytic actions which might be executed, are specified according to the descriptors summarized in Table 2.

The objects O describe the nature of the data to be analyzed and the elements from which the rule-based expert systems infers knowledge. Operations Op establish binary relationships between facts. Thresholds T_h are delimitations

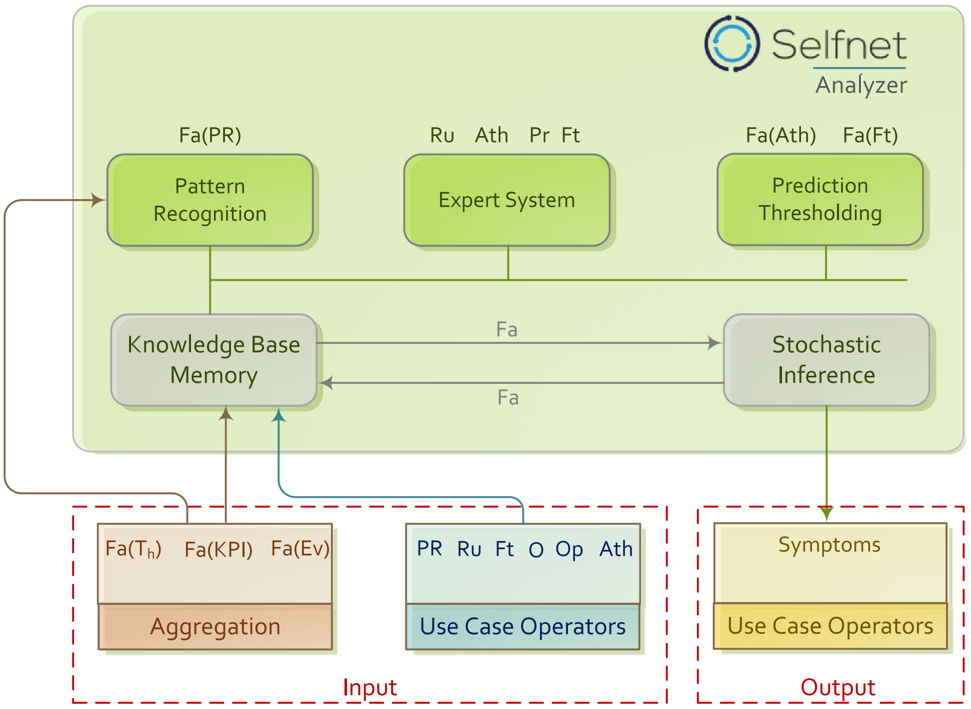


Fig. 3 Inputs and outputs on SELFNET analyzer

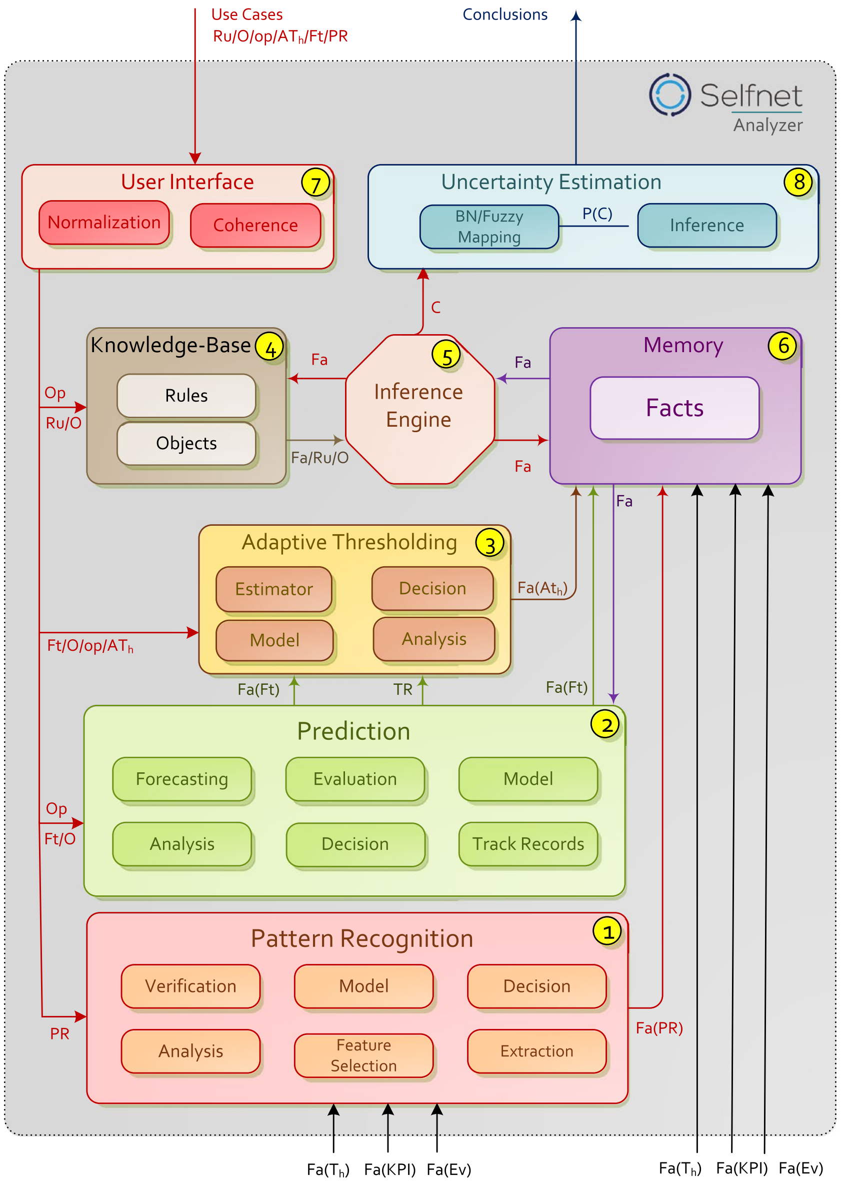


Fig. 4 SELFNET analyzer architecture (Barona López et al. 2017b)

calculated at the Aggregation layer. Facts *Fa* are basic elements of the SELFNET reasoning which describe how the Analyzer Module acquires new knowledge via its rule-based expert system. Rules *Ru* indicate how the SELFNET Analyzer infers new facts at the rule-based expert systems. Note that they are propositional logic expressions in *modus ponens* where the implications deduce the new knowledge. Forecast *Ft*, pattern recognition *PR* and adaptive thresholds AT_h specify the basic analytical operations, for which the datasets *D* provide additional collections of reference samples. Finally, conclusions *C* state facts related with symptoms.

3 Assumptions

The orchestration of the Analyzer accepts the assumptions and limitations established by Barona López et al. (2017b), which were described in the previous section. In order to satisfy the needs of the previously agreed design, as well as to be able to provide the functionalities expected by the rest of the SELFNET tasks, it additionally identifies the following new specific constraints to be considered.

Table 2 Summary of use case data specification

Data	Category	Provider	Destination	Format
Object (simple) O	Specification	Use case	Analyzer	$O_i\colon\{object\ name \vert weight \vert noValues \vert range\ of\ values\ Va\}$
Object (mult) O	Specification	Use case	Analyzer	$O_i\colon\{Object\ name \vert weight \vert noValues \vert \left[Va_1\right]\left[Va_2\right]\ldots[Va_K]\}$
Operation Op	Specification	Use case	Analyzer	$Op_i\colon\{name \vert symbol \vert priority \vert operands \vert description\}$
Facts Fa	Assessment	Aggregation analyzer	Analyzer	$Fa_i\colon\{expression \vert eight \vert uncertainty \vert timestamp \vert location\}$
Rule Ru	Specification	Use case	Analyzer	$Ru_i\colon\{rule \vert priority \vert use\ case\}$
Forecast (ts) Ft	Specification	Use case	Analyzer	$Ft_i\colon\{timeSeries \vert object \vert domain \vert lenght\}$
Forecast (G) Ft	Specification	Use case	Analyzer	$Ft_i\colon\{graph \vert object \vert noVertex \vert domain \vert lenght\}$
Threshold T_h	Specification	Use case	Analyzer	$T_{hi}\colon\{Th\ name \vert object\}$
A. Threshold AT_h	Specification	Use case	Analyzer	$Ft_i\colon\{ATh\ name \vert\ data\ structure \vert CI \vert forecast\}$
Datasets D	Specification	Use case	Analyzer	$D_i\colon\{D\ name \vert object \vert type \vert source\}$
Pattern recognition	Specification	Use case	Analyzer	$PR_i\colon\{PR\ name \vert objectIn \vert objectOut \vert action \vert reference\ data\}$
Conclusion C	Specification	Use case	Analyzer	$St_i\colon\{C\ name \vert use\ case \vert fact\}$
Report Re	Report	Analyzer	Diagnosis	$Re_i\colon\{C\ name \vert use\ case \vert fact \vert uncertainty \vert trigger\}$

3.1 Symptoms and events

The Diagnosis layer of SELFNET (Neves et al. 2016) distinguishes two groups of reports: symptoms and events (Barona López et al. 2017a, b). The first one contains conclusions generated through analytics. On the other hand, events are signals on which it is not necessary to carry out actions related to Artificial Intelligence, such as pattern recognition, prediction or logical inference. Note that in Barona López et al. (2017b), events were managed as facts [in particular $Fa(Ev)$]. In the same way as the rest of the metrics extracted from the aggregated information (see Fig. 4), events were included in the working memory, and hence they could be considered for acquiring knowledge via rule-based expert system, forecasted or studied by pattern recognition techniques. Obviously this was a potential contradiction that must be clarified. In the remainder of this paper, it is assumed that the expression $Fa(Ev)$ strictly refers to aggregated metrics extracted from the monitored events, instead of the event themselves. For example, alerts issued by the IDS involved in the use case Self-Protection are, by definition, events. Given their relevance, they must be directly addressed to the Diagnosis layer, so it is not possible to assume the cost in time that involves the execution of complex analytical calculations on them. However, it is possible to generate metrics that facilitate the making of future decisions or even foresee the issuance of new alerts. For example, the Aggregation layer may provide information about the number of alerts per observation, mean, variance, emission intervals, and its distribution, among others. From which stronger conclusions could be inferred. Unlike when dealing with events, these metrics are not processed with enough efficiency to deliver real-time results.

3.2 Rule based inference

Given the SELFNET framework and the nature of the monitored data, the decision to implement a rule-based inference engine as a symptom discovery tool brings many benefits, among them: (1) rule engines allow to use case administrators decide “What to do”, not “How to do it”. Because of this, it makes it easy to express solutions to difficult problems and specify the onboard of future use cases. (2) It brings logic and data separation, where data is in the domain of objects, and the logic is in the rules Ru. (3) It provides centralization of the knowledge required for infer symptoms. (4) Rule-based systems are fast and scalable: some algorithms (ex. RETE, Leaps, Treat, etc.) (Bassiliades and Vlahavas 1997) and their optimizations (Guillaume and Charnomordic 2012) provide very efficient ways of matching rule patterns to the use cases domain object data. These are especially efficient when facts change in small portions as the rule engine can remember past matches. For example, this happens with the information periodically provided by a particular SELFNET sensor. But rule-based systems also pose drawbacks: the first of them is high dependency of the rule set. If the rules are not consistent, coherent or reasonably specific, the results obtained will be probably not as expected (Lunardhi and Passino 1995). On the other hand, they are susceptible to bad practices. For example, rule-based systems allow storing, managing and updating rules as data. It is common that they are mistakenly used to generate new rules or even update them at runtime, which is out of the scope of these technologies. Finally, it is important to bear in mind that the scalability of rule-based systems has a negative impact in terms of resource consumption. In this regard, it is worth mentioning the consequences of their two most frequent ways to scale (Wang and Hanson 1992): firstly, if the number of facts is acceptable, but the number of rules is very high, there will be an important increase in the computation time of their processing. On the opposite, if the number of facts is very high, but the number of rules is acceptable, a larger amount of memory is required for storage. Note that if the number of inputs and rules are large, then both, memory and efficiency are penalized. In the context of SELFNET it is expected to receive a large number of facts, but operate on small rule sets. Consequently, it is expected that the scalability of the expert rule-based system will lead to the use of a greater amount of storage space.

3.3 Data granularity

SELFNET is a complex monitoring scenario where a large amount of sensors collect information about the state of the network in real time. This information is processed in the aggregation layer, which provides the necessary metrics to acquire knowledge. For this purpose, the Analyzer must perform complex calculations. As will be described in the later sections, aggregated data will not be raw processed. Instead, it will be packed as Aggregated Data Bundles (ADB) which will periodically be loaded by the Analyzer and converted into facts. Each ADB is the summary of all the system information observed over a time period T. It can therefore be stated that ADB may be abstracted as an observation on a time series of records that facilitate the network awareness. It is assumed that the effectiveness and performance of the analytics depends on the T, and how representative is the information on the ADB.

4 Design principles

The following design principles and limitations lay the foundation of the Analyzer orchestrator, as well as the implementation of its internal components, data flows and synchronization.

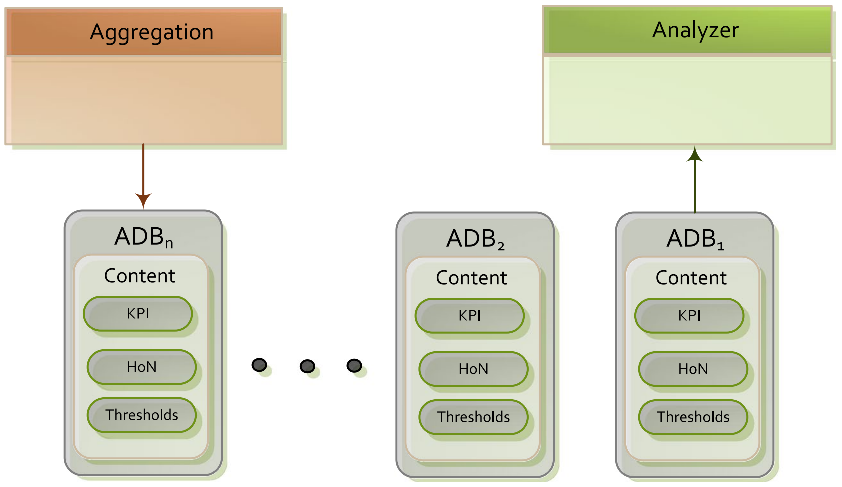
4.1 Aggregated data bundles

The information required for the analytics is obtained from the Aggregation layer packaged as Aggregated Data Bundles (ADB). An ADB is the summary of the aggregated metrics calculated in a time interval P translated into facts Fa. Note that a priori, the data within an ADB does not overlap the metrics on other ADBs (this aspect could be revised later for future optimizations). For example, let the time series $Y = \{Y_t : t \in T\}$ where $Y_1, Y_2, \ldots, Y_k$, $k = 7$, assuming the construction of ADBs on $P = 1$, the SELFNET Analyzer will sequentially deal with seven ADBs, i.e. $ADB_1, ADB_2, \ldots, ADB_k$ (see Fig. 5). Through the use of this strategy a massive and continuous input of information is avoided, which facilitate the initialization of the implemented data mining algorithms. Likewise, the information is managed and processed in an orderly manner, which also reduces the number of inconsistencies between the new facts and the data stored in the working memory. Finally, as is illustrated at the next section, the deployment of optimization method based on the exploitation of concurrence is facilitated.

4.2 Persistence

The SELFNET Analyzer does not provide persistence of the data loaded as ADBs. The monitored raw data and aggregated metrics are conveniently stored in the Big Data platform located at the Aggregation layer. Facts Fa not implicated in prediction/pattern recognition are discarded once their ADB is completely processed and the conclusions are inferred. This means that, in this case, facts Fa are temporally stored in a local short-term memory only for the duration of their analysis. On the other hand, facts Fa

Fig. 5 Communication by ADBs



required for prediction/pattern recognition may temporally persist throughout the analysis of various ADBs. This is because they compose the time series and graphs needed to build models/regressions. Note that these data structures have limited size, which once reached involves eliminating the more obsolete observations via First In First Out (FIFO) policies (Finkel et al. 2003). Once an ADB is completely analyzed and the conclusions are reported to the Diagnosis layer as symptoms, the working memory of the rule-based expert systems is restarted. Only the necessary facts for the construction of the time series and graphs are temporarily conserved, but this is outside the working memory. When loading a new ADB, facts on time series and graphs are again, added to the working memory as *Fa* (*Ft*), *Fa* (*Ath*) and *Fa*(*PR*).

4.3 Analytic pipelining

Analytics are executed as a linear pipeline of sets of data processing elements connected in series, where the output of an input is the input of the next one (Zou et al. 2014). When an ADB reach the SELFNET Analyzer, a sequence of processing elements is executed, where intelligence actions (i.e. logic inference, pattern recognition, prediction) and preprocessing steps (load ADBs, data encapsulation, generation of reports) are chronologically separated, and their inputs/outputs are shared by buffer storage structures. So it is possible to state that this first approach considers a buffered-synchronous pipeline analytic architecture. Its main advantages are: great organization of information to process, mitigation of inconsistencies between the new facts and the data being analyzed, easy of design and modularity. The latter allows managing every set of actions independently, which facilitates debugging, troubleshooting tasks and provide a more accurate assessment of the performance of their analytic actions. But it is important to keep in mind that this scheme also poses several challenges, among them try to define sets of actions of similar complexity in order to enable optimization strategies based on parallelism, the fact that the delay in a task may slow down the execution of those that depend on it, and in the case of implement parallelism, the best suited politics of temporal memory sharing must be identified.

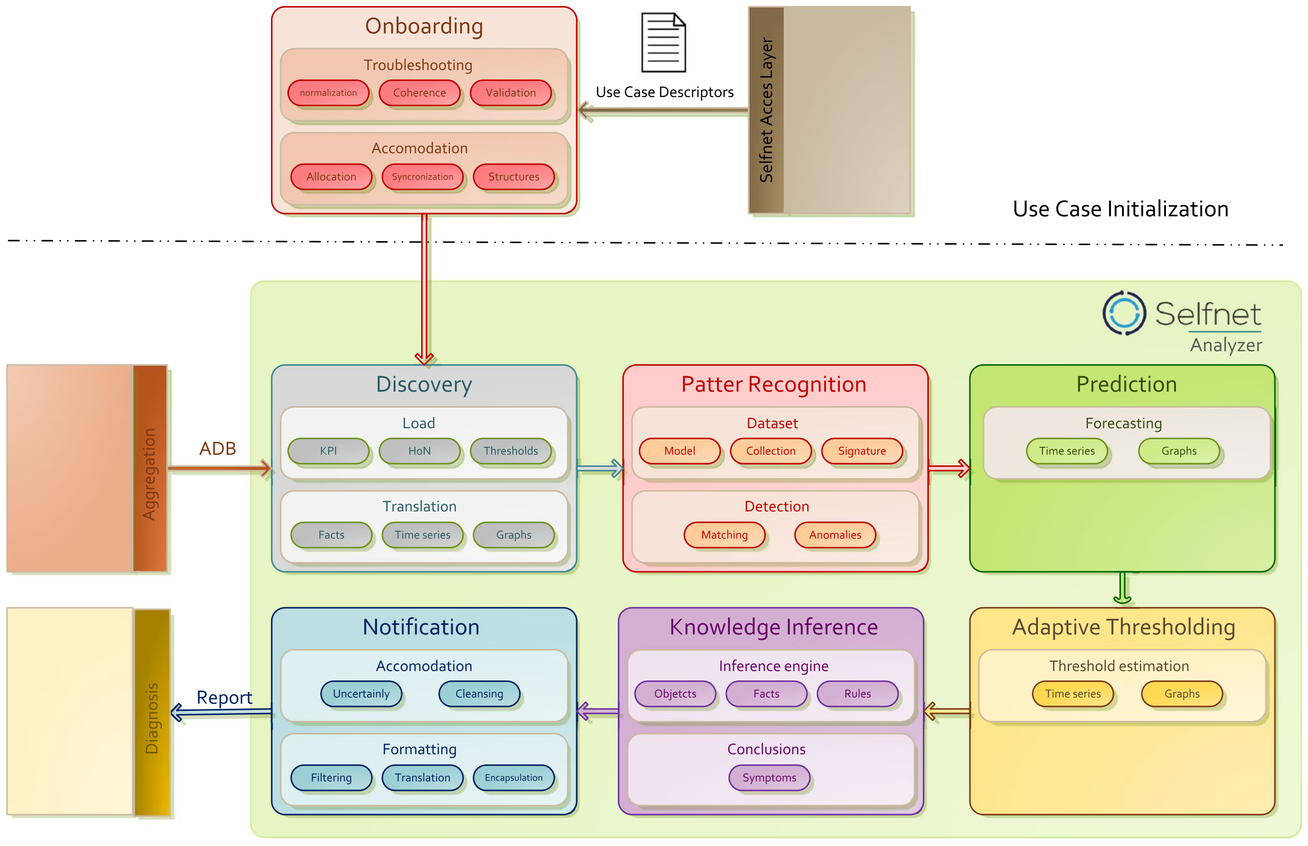


Fig. 6 Sets of actions on the analyzer

5 Workflow

The SELFNET Analyzer orchestration is separated into seven main steps: use case Onboarding (O), Discovery (DIS), Patter Recognition (PR), Prediction (FT), Adaptive Thresholding (ATH), Knowledge inference (KI) and Notification (N). They are illustrated in Fig. 6 and described in detail below.

- *Onboarding [O]* The onboarding step is executed only once per use case. It corresponds to the component User Interface in Barona López et al. (2017b), and allows updating the knowledge-base by inserting, modifying or deleting data associated with every use case, such as objects O, rules Ru operations Op or prediction metrics Ft. When a new use case is onboarded, the input data is normalized, and in order to avoid runtime errors, the coherence of the new specification is validated. Then the Analyzer is prepared to accommodate the new operations, hence including the specified information on the existing data structures, memory allocation and synchronization of the onboarded actions with the previous loaded configurations.
- *Discovery [DIS]* The discovery step is the link between the SELFNET Aggregation and Analyzer layers. These tasks periodically receive ADBs which summarize the SELFNET aggregated observations. From the loaded KPI, events and thresholds, the Analyzer build facts ($Fa(KPI)$, $Fa(Ev)$ and $Fa(Th)$). If they are required for prediction, patter recognition or adaptive thresholding, the Analyzer includes these observations in the temporally stored time series or graphs. Note that independent facts are removed at the end of the ADB processing, as well as the new knowledge acquired from them.
- *Pattern recognition [PR]* The set of actions related with pattern recognition implies the access to the datasets with models, sample collection or signatures, and the detection of matches or outliers. The acquired facts may be considered by prediction, pattern recognition or adaptive thresholding, as well as to infer knowledge on the rule-based expert system.
- *Prediction [FT]* The set of actions related with prediction includes the construction of forecasting models/regression, the decision of the best suited algorithms by considering the nature of the input data, and the estimation of its evolution. As is the case on the pattern recognition activities, the generated facts may be considered to infer knowledge on the rule-based expert system, and also to identify adaptive thresholds.
- *Adaptive thresholding [ATH]* This set of operations establishes measures to approximate when the forecasting errors must be taken into account when identifying symptoms. In order to enhance the information reported to the Diagnosis layer, the new facts are provided to the rule-based expert system, hence contributing to the inference of new knowledge.
- *Knowledge inference [KI]* This step executes the tasks related with the rule-based expert system. It considers the data provided by the sources of information mentioned above, among them facts directly built from aggregated data, pattern recognition, prediction and adaptive thresholding steps. The acquired knowledge is included in the SELFNET Analyzer working memory. Conclusions are transmitted to the notification capabilities as potential symptoms.
- *Notification [N]* The set of actions on Notification corresponds to those on the component Uncertainty Estimation at the original SELFNET Analyzer architecture. They are the link between the SELFNET Diagnosis layer and the knowledge acquired by the Analyzer. This step performs two main groups of tasks: accommodation and formatting. The first one filter redundant and low representative information. Once the ADB is completely analyzed, these actions erase and restart the auxiliary functionalities on the analytics and the several data structures; only the information required for build time series and graphs from data included in future ADBs is temporally persistent. On the other hand, the group of actions related with formatting, translates internal information of the analyzer to crisp data required by Diagnosis. Then it is reported.

6 Execution and optimization

When no optimization measures are implemented, the execution of the sets of actions determined in the previous section can be summarized in Fig. 7. There the onboard of a new use case and the completion of its different task are illustrated. Note that in accordance with this basic specification, the Analyzer only is able to load a new ADB if the previously loaded ADB is completely processed. Obviously this is not the most efficient way to carry out their study. Assuming separately the computational costs of every set of actions: O(DIS), O(PR), O(FT), O(ATH), O(FT), O(KI) and O(N); and ignoring the penalty of onboarding use cases O($Onboard$), the average cost of analyze an ADB is:

$$O_{ADB} = O(DIS + PR + FT + ATH + KI + N)$$

Where given the complexity of the pattern recognition and prediction methods, O(FT) and O(ATH) will concentrate most of the resource penalty. This approach is cheap in terms of memory, because once a set of actions is completed, most of their auxiliary data structures can be released. In addition, not managing different ADBs in parallel prevents the replication of such containers. Because of

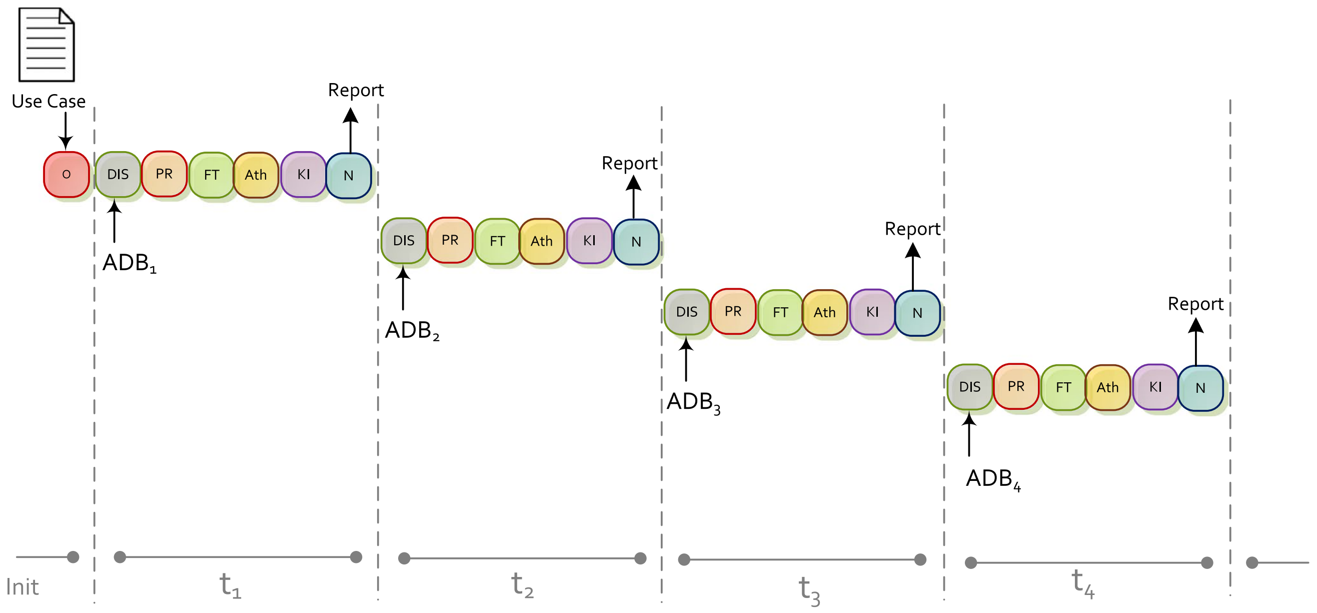


Fig. 7 Basic execution of the analyzer sets of actions

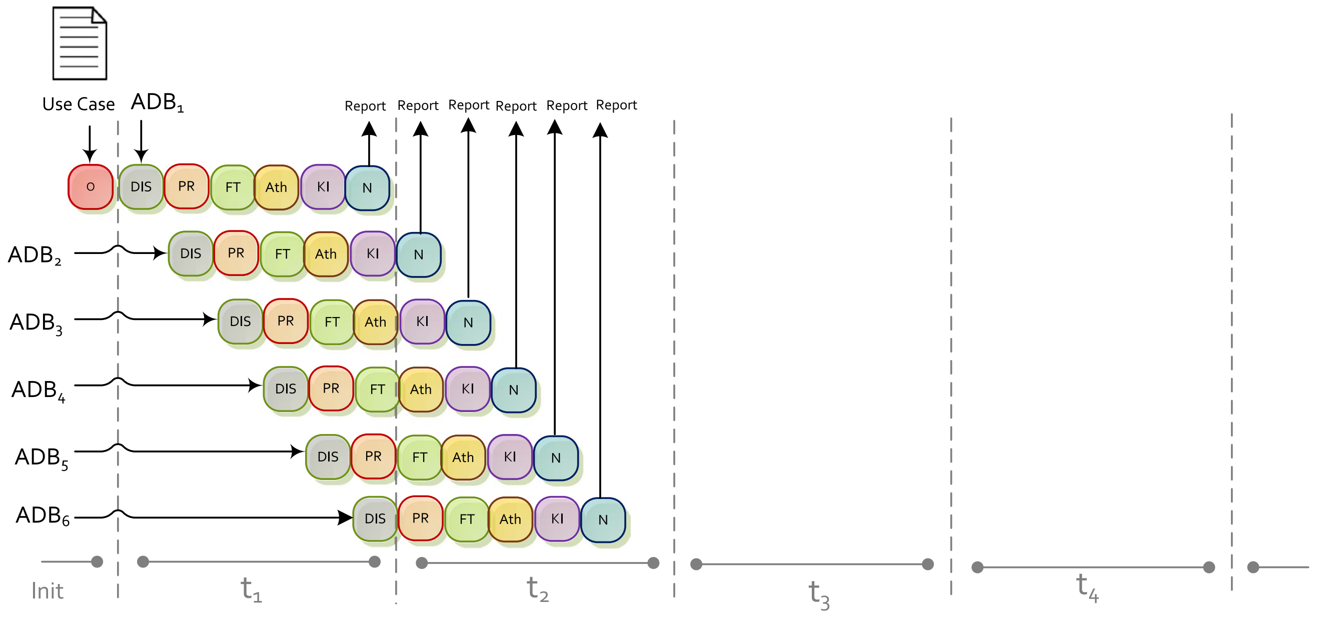


Fig. 8 Example of optimal analysis of multiple ADBs in concurrency

its simplicity and easy debugging, this is the first version of the Analyzer orchestrator that was implemented.

However, this scheme can be optimized easily by considering pipelining solutions. They allow overlapping execution of multiple actions with the same memory space by exploiting parallelism (Gordon et al. 2006). Figure 8 illustrates an example of these kinds of methods, where six ADBs can be processed at the same time period.

Two sets of similar actions cannot be processed in concurrency, but it is possible with different sets. This means that, for example Patter Recognition on the analysis of ADB_1 cannot overlap with Pattern Recognition on the following ADB_2, since all the resources for this task are being used to analyze the first information package. But it could be executed in concurrency with the Discovery stage of ADB_2, where resources and memory are not shared. If the initialization cost related with processing the first ADB is ignored, it can be formalized as follows:

$$O_{init} = O(DIS + PR + FT + ATH + KI + N)$$

Then the cost of analyze ADB_s at $init + 5$ is summarized as

$$O_{ADB} = O(MAX\{DIS, PR, FT, ATH, KI, N\})$$

This implies an important improvement over the original proposal. But the implementation of this scheme leads to several restrictions. Firstly, it requires a greater amount of memory; the system must support up to six times more storage space to facilitate the analysis of six ADBs at a time. On the other hand, in order to allow the communication between sets of actions, the SELFNET Analyzer must provide temporal storage buffers and synchronization mechanism. This requires managing shared memory between tasks, and adds complexity to the execution thread. Furthermore, it has to be borne in mind that under optimal circumstances, all sets of actions must take the same time to complete. Obviously this does not happen in reality, since pattern recognition and prediction actions often imply a higher cost than those relate with the rule-based inference. Consequently, it is possible that certain sets of actions must remain on hold until others are finished, before giving way to new analysis processes. This problem is illustrated in Fig. 9, where the different sets of actions display unequal time consumption. If there are no waits, the different tasks will overlap leading to memory-sharing conflicts and inconsistencies between facts. For example, ADB_3 prediction actions require the pattern recognition facts of the same processing thread. But if such overlapping occurs, prediction on ADB_3 may also receive facts derived from pattern recognition at ADB_2, which would lead to inference erroneous knowledge. Note that it is also possible that none of the use cases require the execution of some sets of actions (in Fig. 9 this occurs with adaptive thresholding tasks). In both circumstances there will be moments of waiting.

Another clear example of inequality between execution costs of sets of actions is shown in Fig. 10, when the same task, in this case prediction, becomes more and more expensive over time. This entails an accumulative delay in the previous actions (pattern recognition).

The Analyzer orchestrator deals with these problems by adjusting the granularity of the information provided by the ADBs, and by limiting the observation sliding windows and the amount of information considered for initializing the pattern recognition and prediction algorithms. According to these circumstances, the cost of executing an ADB_s at $init + 5$ once the sequencing is initialized is expressed as follows:

$$O_{ADB} = O(MAX\{DIS, PR, FT, ATH, KI, N\}) + delay_t$$

Where the cumulative penalization is decomposed as:

$$delay_t = O\big(Wait_{DIS} + Wait_{PR} + Wait_{FT} + Wait_{ATH} + Wait_{KI} + Wait_N\big)$$

Alternatively, each set of actions is also able to exploit concurrency at thread level in order to improve its performance. Due to the characteristics of the monitoring environment, it is possible to deduce that frequently, the same

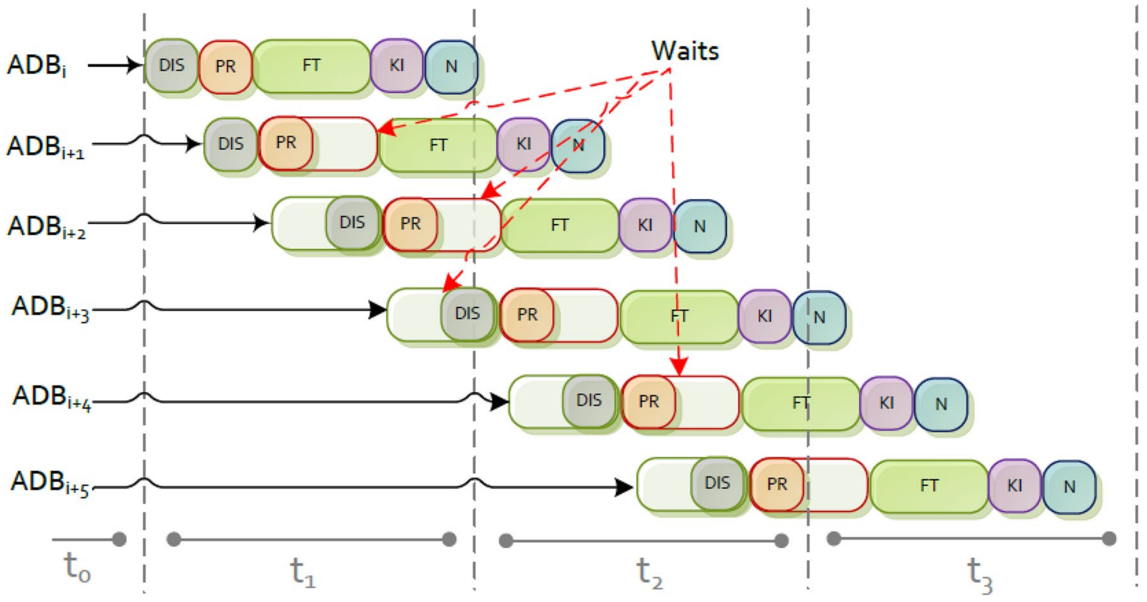


Fig. 9 Example of computational time penalization because of unequal set of actions

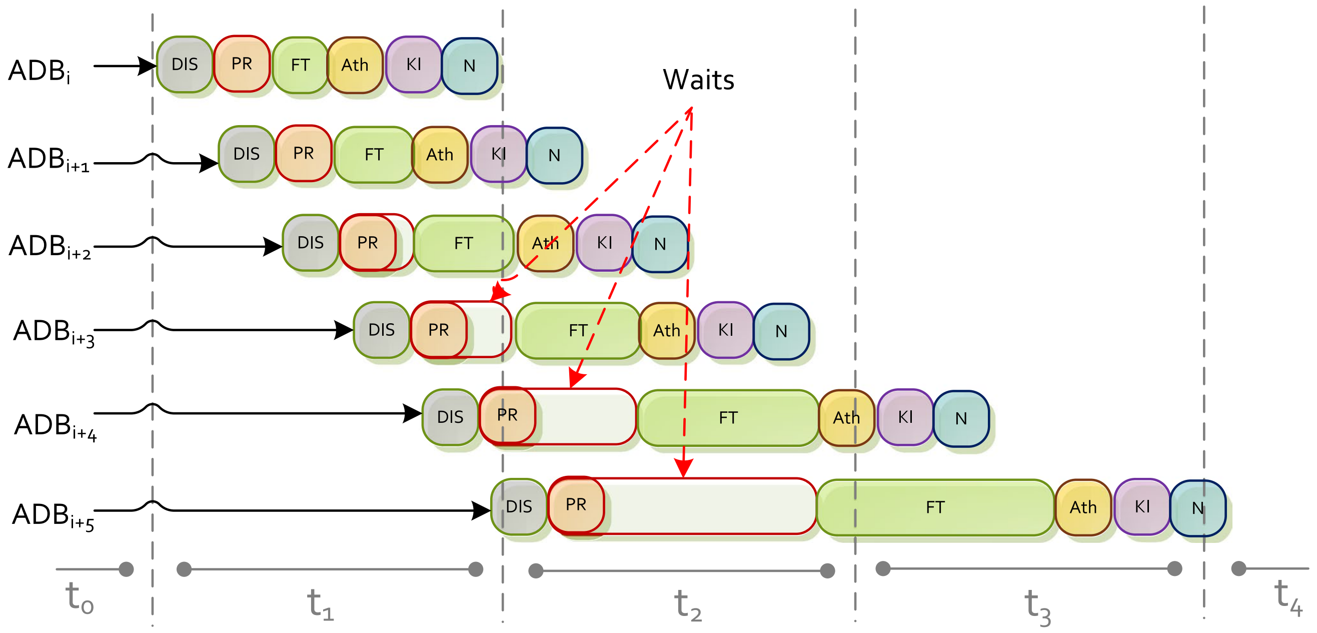


Fig. 10 Example of computational time penalization because of incremental resource consumption

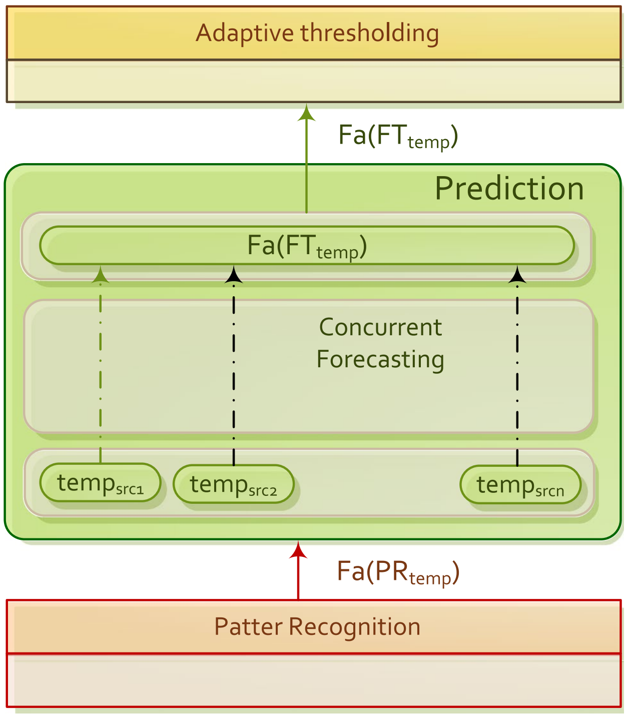


Fig. 11 Example of concurrency exploitation

Table 3 *SLP* onboarding specification

Item	Descriptor
Object	O_1:{*Packetloss*|1|1|$\mathbb{R}$}
Threshold	T_{h1}:{max *PacketLoss*|O_1}
Operator	Op_1:{*Equal*| = | 1 |(*Fa*, *O*, *Va*) = (*Fa*, *O*, *Va*)|*equal*}
Operator	Op_2:{*LGT*| ⩾ | 1 |(*Fa*, *O*, *Va*) ⩾ (*Fa*, *O*, *Va*)|*left is GE*}
Conclusion	C_1:{*excessive packet loss*|*SLP*|*Fa*(O_1) ⩾ *Fa*(T_{h1})}
Rule	Ru_1:{*Fa*(O_1) ⩾ *Fa*(T_{h1}) → *Fa*(C_1)|1|*SLP*}

metric (ex. temperature, congestion, etc.) may be reported from different sources. The analysis of similar information, but provided from different data sources, is enhanced by CPU/GPU multithreading (Su et al. 2016) as it is illustrated in Fig. 11, which improves their consumption of computational resources in terms of storage and efficiency.

7 Illustrative examples

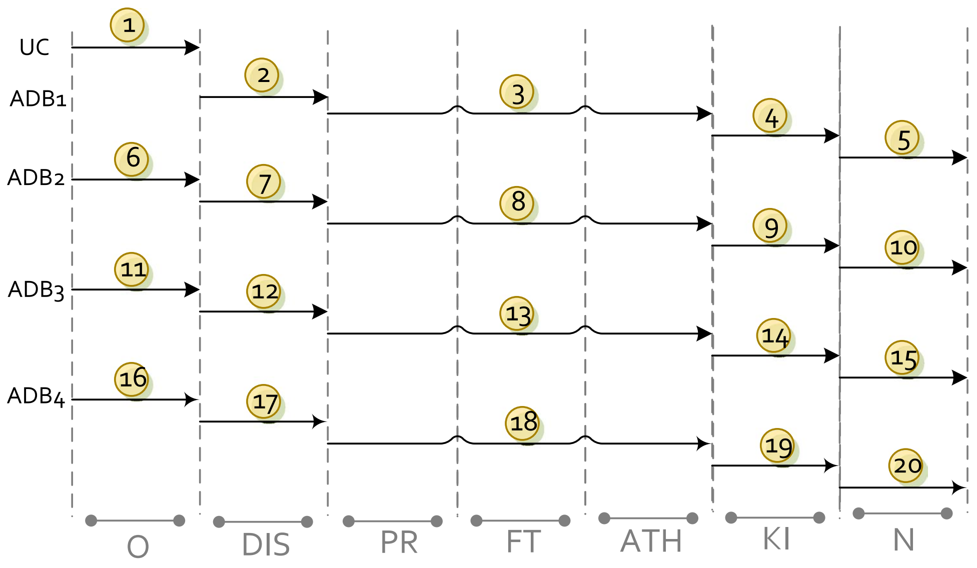
This section describes several examples of the analytic process according with the aforementioned Analyzer Orchestration scheme.

7.1 UC 1: device packet loss

7.1.1 Description

The illustrative use case called *Self—packet loss prevention* (SLP) reports symptoms related with huge packet loss rates on SELFNET devices. Where if the packet loss rate of certain SELFNET device exceeds a specific threshold, a new fact that represents such situation is acquired. This is a very

Fig. 12 Example of runtime in self-packet loss prevention



basic example where concurrency pipelining is not applied, and where prediction and adaptive thresholding are not considered. Therefore the decision thresholds are static and were built at Aggregation. Table 3 shows its onboarding descriptors according with the specification summarized in Table 2.

7.1.2 Step-by-step

The following illustrates and example of runtime in *Self—packet loss prevention*, where different ADBs are loaded and analyzed according to the aforementioned indications. Figure 12 displays every step in a sequence diagram, which are described step-by-step below:

1. The *SLP* use case descriptors are loaded by the SELFNET Analyzer. Then, the memory for storing temporal containers of objects O_1, thresholds T_{h1} and facts Fa is allocated. Pattern recognition, prediction and adaptive thresholding are not required, so neither data structures to support time series nor graphs are considered.
2. The ADB_1 with aggregated instances of O_1 and T_{h1} is requested to the Aggregation layer and then it is processed. The SELFNET Analyzer build the following facts from the information gathered by the network elements (*NodeA*, *NodeB*, *NodeC*, *NodeD*):

   $Fa(O)[idO1]: \{O_1 = 0.35|1|1|Today\ 12{:}22{:}15|NodeA\}$
   $Fa(O)[idO2]: \{O_1 = 0.34|1|1|Today\ 12{:}22{:}15|NodeB\}$
   $Fa(O)[idO3]: \{O_1 = 0.33|1|1|Today\ 12{:}22{:}15|NodeC\}$
   $Fa(O)[idO4]: \{O_1 = 0.35|1|1|Today\ 12{:}22{:}15|NodeD\}$

   And by the data Aggregation:

   $Fa(Th)[idTh1]: \{T_{h1} = 0.7|1|1|Today\ 12{:}22{:}15|All\}$
3. Given that pattern recognition, prediction and adaptive thresholding are not required, the Analyzer bypasses those steps (i.e. new facts are not inferred by them).
4. The rule-based inference engine processes the discovered facts by applying the rule Ru_1. Given that the condition $Fa(O_1) \geqslant Fa(T_{h1})$ is not satisfied by any of the facts, conclusions related with *SLP* are not inferred.
5. All the temporal data related with objects O_1, thresholds T_{h1} and facts Fa is cleaned.
6. The ADB_2 with aggregated instances of O_1 and T_{h1} is requested to the Aggregation layer and processed. The SELFNET Analyzer build the following facts from the information gathered by the network elements (*NodeA*, *NodeB*, *NodeC*, *NodeD*):

   $Fa(O)[idO5]: \{O_1 = 0.36|1|1|Today\ 12{:}23{:}15|NodeA\}$
   $Fa(O)[idO6]: \{O_1 = 0.34|1|1|Today\ 12{:}23{:}15|NodeB\}$
   $Fa(O)[idO7]: \{O_1 = 0.81|1|1|Today\ 12{:}23{:}15|NodeC\}$
   $Fa(O)[idO8]: \{O_1 = 0.31|1|1|Today\ 12{:}23{:}15|NodeD\}$

   And by the data Aggregation:

   $Fa(Th)[idTh2]: \{T_{h1} = 0.79|1|1|Today\ 12{:}23{:}15|All\}$
7. The Analyzer bypasses pattern recognition, prediction and adaptive thresholding.
8. The rule-based inference engine processes the discovered facts by applying the rule Ru_1. Given that the condition $Fa(O_1) \geqslant Fa(T_{h1})$ is satisfied for the data gathered by *NodeC*, The following fact related with $Self - packet\ loss\ prevention$ is inferred.

$Fa[idF1]$: $\{Fa(idO7) \geqslant Fa(idTh2)|1|1|Today\ 12{:}23{:}15|NodeC\}$.

Which describes the conclusion C_1:

$C_1[idC1]$:$\{excessive\ packet\ loss|SLP|Fa(idO7) \geqslant Fa(idTh2)\}$

9. The following symptom is reported to the Diagnosis layer:

   $Re_1[idRe1]$:

   $\{excessive\ packet\ loss|SLP|idF1|1|idO7, idTh2, Ru_1\}$

   All the temporal data related with objects O_2, thresholds T_{h2} and facts Fa is cleaned.
10. The ADB_3 with aggregated instances of O_1 and T_{h1} is requested to the Aggregation layer and processed. The SELFNET Analyzer builds the following facts from the information gathered by the network elements (*NodeA*, *NodeB*, *NodeC*, *NodeD*):

    $Fa(O)[idO09]$: $\{O_1 = 0.36|1|1|Today\ 12{:}24{:}15|NodeA\}$
    $Fa(O)[idO10]$: $\{O_1 = 0.34|1|1|Today\ 12{:}24{:}15|NodeB\}$
    $Fa(O)[idO11]$: $\{O_1 = 0.33|1|1|Today\ 12{:}24{:}15|NodeC\}$
    $Fa(O)[idO12]$: $\{O_1 = 0.31|1|1|Today\ 12{:}24{:}15|NodeD\}$

    And by the data Aggregation:

    $Fa(Th)[idTh3]$: $\{T_{h1} = 0.77|1|1|Today\ 12{:}24{:}15|All\}$
11. The Analyzer bypasses pattern recognition, prediction and adaptive thresholding.
12. The rule-based inference engine processes the discovered facts by applying the rule Ru_1. Given that the condition $Fa(O_1) \geqslant Fa(T_{h1})$ is not satisfied by any of the facts, conclusions related with *SLP* are not inferred.
13. All the temporal data related with objects O_1, thresholds T_{h1} and facts Fa is cleaned.

7.2 UC 2: quality of service analysis

7.2.1 Description

The illustrative use case to be managed *Self—QoSOverwatch* (SQoS) report symptoms related with suspicious QoS decreasing, In particular, if a significantly decrement considering the latest observations is detected, a new fact related with relevant QoS variation is acquired. In this context, concurrency at pipelining is not applied, prediction and adaptive thresholding are considered, and it is assumed that the forecasting algorithm requires at least $n = 8$ observations for building the prediction model. Table 4 shows its onboarding descriptors according with the specification summarized in Table 2.

7.2.2 Step-by-step

The following illustrates and example of runtime in *Self—QoSOverwatch*, where different ADBs are loaded and analyzed according to the aforementioned indications. Figure 13 displays every step in a sequence diagram, which are described step-by-step below:

1. The descriptors of the *SQoS* use case are loaded by the SELFNET Analyzer. Then, the memory for storing temporal containers of objects O_1, forecasts F_{t1}, adaptive tresholds AT_{h1} and facts Fa is allocated. Prediction capabilities on time series are required, so the data structures to support time series are initiated.
2. The ADB_1 with aggregated instances of O_1 is requested to the Aggregation layer and then it is processed. The SELFNET Analyzer built the following facts from the information gathered by the network elements (*NodeA*, *NodeB*, *NodeC*, *NodeD*):

   $Fa(O)[idO1]$: $\{O_1 = 0.60|1|1|Today\ 12{:}22{:}15|NodeA\}$
   $Fa(O)[idO2]$: $\{O_1 = 0.65|1|1|Today\ 12{:}22{:}15|NodeB\}$
   $Fa(O)[idO3]$: $\{O_1 = 0.61|1|1|Today\ 12{:}22{:}15|NodeC\}$
   $Fa(O)[idO4]$: $\{O_1 = 0.62|1|1|Today\ 12{:}22{:}15|NodeD\}$
3. Given that the Analyzer does not dispose of time series of $n = 8$ facts per sensor, prediction is not possible. Hence, adaptive thresholding is not performed. Because there are not facts related with adaptive thresholds, the rule Ru_1 where $Fa(O_1) \geqslant Fa(AT_{h1}) \rightarrow Fa(C_1)$ cannot be triggered. So conclusions related with symptoms are not notified to the diagnosis layer. On the other hand, given that the acquired facts are related with time series analysis (i.e. prediction and adaptive thresholding), they cannot be deleted before load-

Table 4 *Self—QoSOverwatch* specification

Item	Descriptor
Object	O_1:$\{QoS\ decrement|1|1|[0,1]\}$
Forecast	Ft_1:$\{timeSeries|O_1|obs|t+1\}$
Adaptive threshold	AT_{h1}:$\{maxQoS\ decrement|timeSeries|0.95|Ft_1\}$
Operator	Op_1:$\{Equal| = |1|(Fa, O, Va) = (Fa, O, Va)|equal\}$
Operator	Op_2:$\{LGT| \geqslant |1|(Fa, O, Va) \geqslant (Fa, O, Va)|left\ is\ GE\}$
Conclusion	C_1:$\{Suspicious\ QoS\ variation|SQoS|Fa(O_1) \geqslant Fa(AT_{h1})\}$
Rule	Ru_1:$\{Fa(O_1) \geqslant Fa(AT_{h1}) \rightarrow Fa(C_1)|1|SC\}$

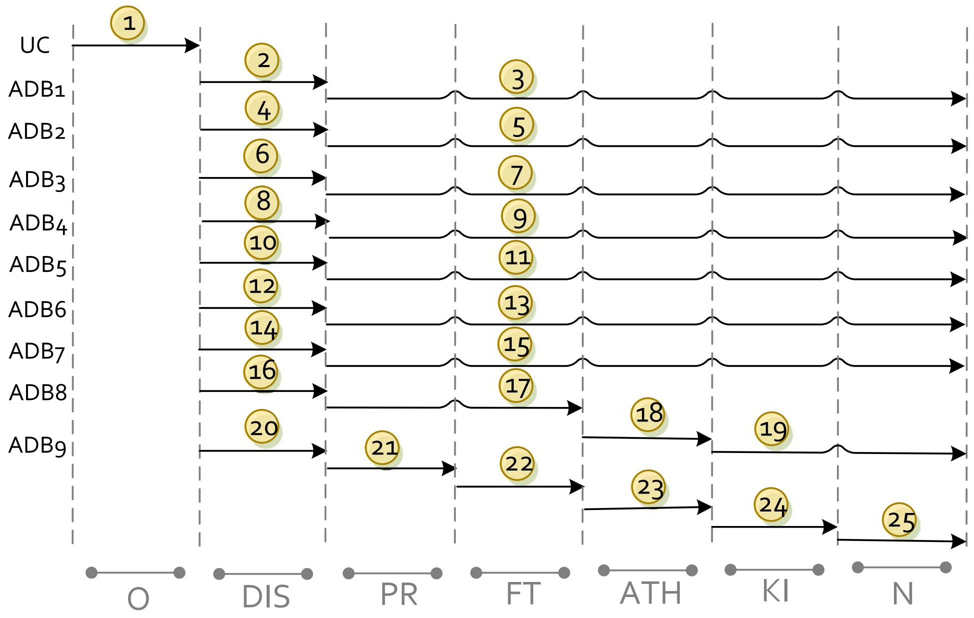


Fig. 13 Example of runtime in Self-QoSOverwatch

Table 5 Facts ADB_2 to ADB_7

ADB	Facts
ADB_2	$Fa(O)[idO5]\{O_1 = 0.63\|1\|1\|Today$ 12:23:15$\|NodeA\}$ $Fa(O)[idO6]\{O_1 = 0.64\|1\|1\|Today$ 12:23:15$\|NodeB\}$ $Fa(O)[idO7]\{O_1 = 0.65\|1\|1\|Today$ 12:23:15$\|NodeC\}$ $Fa(O)[idO8]\{O_1 = 0.66\|1\|1\|Today$ 12:23:15$\|NodeD\}$
ADB_3	$Fa(O)[idO9]\{O_1 = 0.62\|1\|1\|Today$ 12:24:15$\|NodeA\}$ $Fa(O)[idO10]\{O_1 = 0.70\|1\|1\|Today$ 12:24:15$\|NodeB\}$ $Fa(O)[idO11]\{O_1 = 0.72\|1\|1\|Today$ 12:24:15$\|NodeC\}$ $Fa(O)[idO12]\{O_1 = 0.63\|1\|1\|Today$ 12:24:15$\|NodeD\}$
ADB_4	$Fa(O)[idO13]\{O_1 = 0.60\|1\|1\|Today$ 12:25:15$\|NodeA\}$ $Fa(O)[idO14]\{O_1 = 0.72\|1\|1\|Today$ 12:25:15$\|NodeB\}$ $Fa(O)[idO15]\{O_1 = 0.73\|1\|1\|Today$ 12:25:15$\|NodeC\}$ $Fa(O)[idO16]\{O_1 = 0.65\|1\|1\|Today$ 12:25:15$\|NodeD\}$
ADB_5	$Fa(O)[idO17]\{O_1 = 0.62\|1\|1\|Today$ 12:26:15$\|NodeA\}$ $Fa(O)[idO18]\{O_1 = 0.71\|1\|1\|Today$ 12:26:15$\|NodeB\}$ $Fa(O)[idO19]\{O_1 = 0.76\|1\|1\|Today$ 12:26:15$\|NodeC\}$ $Fa(O)[idO20]\{O_1 = 0.63\|1\|1\|Today$ 12:26:15$\|NodeD$
ADB_6	$Fa(O)[idO21]\{O_1 = 0.63\|1\|1\|Today$ 12:27:15$\|NodeA\}$ $Fa(O)[idO22]\{O_1 = 0.70\|1\|1\|Today$ 12:27:15$\|NodeB\}$ $Fa(O)[idO23]\{O_1 = 0.71\|1\|1\|Today$ 12:27:15$\|NodeC\}$ $Fa(O)[idO24]\{O_1 = 0.60\|1\|1\|Today$ 12:27:15$\|NodeD\}$
ADB_7	$Fa(O)[idO25]\{O_1 = 0.61\|1\|1\|Today$ 12:28:15$\|NodeA\}$ $Fa(O)[idO26]\{O_1 = 0.72\|1\|1\|Today$ 12:28:15$\|NodeB\}$ $Fa(O)[idO27]\{O_1 = 0.73\|1\|1\|Today$ 12:28:15$\|NodeC\}$ $Fa(O)[idO28]\{O_1 = 0.62\|1\|1\|Today$ 12:28:15$\|NodeD\}$

ing the following ADBs, but the rule-based inference engine is reinitiated.

4. The Analyzer performs the same actions (Steps 2 and 3) from ADB_2 until ADB_7. Table 5 shows the facts built for these set of ADBs.
5. The ADB_8 with aggregated instances of O_1 is requested to the Aggregation layer and then it is processed. The

Table 6 Summary of information on time series at SQoS

Time	N	NodeA	NodeB	NodeC	NodeD
12:22:15	1	0.60	0.65	0.61	0.62
12:23:15	2	0.63	0.64	0.65	0.66
12:24:15	3	0.62	0.70	0.72	0.63
12:25:15	4	0.6	0.72	0.73	0.65
12:26:15	5	0.62	0.71	0.76	0.63
12:27:15	6	0.63	0.7	0.71	0.6
12:28:15	7	0.61	0.72	0.73	0.62
12:29:15	8	0.6	0.73	0.72	0.64
Forecast	n+1	0.61	0.72	0.72	0.63

SELFNET Analyzer built the following facts from the information gathered by the network elements:

$Fa(O)[idO29]$: $\{O_1 = 0.60|1|1|Today$ 12:29:15$|NodeA\}$
$Fa(O)[idO30]$: $\{O_1 = 0.73|1|1|Today$ 12:29:15$|NodeB\}$
$Fa(O)[idO31]$: $\{O_1 = 0.72|1|1|Today$ 12:29:15$|NodeC\}$
$Fa(O)[idO32]$: $\{O_1 = 0.64|1|1|Today$ 12:29:15$|NodeD\}$

6. A this point, there are $n = 8$ facts per sensor in the time series to be predicted, so the forecasting method are able to estimate the next observation $(t + 1)$ as specified in the use case definition. The temporally stored data is summarized in Table 6.
   The following facts related with prediction are acquired:

$Fa(Ft)[idF1]$: $\{Ft_1 = 0.61|1|1|Today$ 12:29:15$|NodeA\}$
$Fa(Ft)[idF2]$:$\{Ft_1 = 0.72|1|1|Today$ 12:29:15$|NodeB\}$
$Fa(Ft)[idF3]$:$\{Ft_1 = 0.72|1|1|Today$ 12:29:15$|NodeC\}$
$Fa(Ft)[idF4]\{Ft_1 = 0.63|1|1|Today$ 12:29:15$|NodeD\}$

7. The following facts related with the adaptive thresholds built from the predictions are acquired:

   $Fa(Ath)[idA1]$: $\{Ath_1 = 0.62|1|1|Today\ 12{:}29{:}15|NodeA\}$
   $Fa(Ath)[idA2]$:$\{Ath_1 = 0.73|1|1|Today\ 12{:}29{:}15|NodeB\}$
   $Fa(Ath)[idA4]\{Ath_1 = 0.64|1|1|Today\ 12{:}29{:}15|NodeD\}$

8. The recent calculated thresholds are not applicable to the current observations, so the rule Ru_1 where $Fa(O_1) \geqslant Fa(AT_{h1}) \rightarrow Fa(C_1)$ cannot be triggered. Conclusions related with symptoms are not noEq210tified to the diagnosis layer.
   Note that for the observation i only the predictions and adaptive thresholds calculated at $0, \dots, i-1$ can be considered; stated in another way: predictions and adaptive thresholds calculated at i are only valid for the next $i+1$ observations, when it can be verified whether they have been fulfilled.

9. The ADB_9 with aggregated instances of O_1 is requested to the Aggregation layer and then it is processed. The SELFNET Analyzer built the following facts from the information gathered by the network elements (*NodeA*, *NodeB*, *NodeC*, *NodeD*):

   $Fa(O)[idO33]$: $\{O_1 = 0.60|1|1|Today\ 12{:}30{:}15|NodeA\}$
   $Fa(O)[idO34]$: $\{O_1 = 0.82|1|1|Today\ 12{:}30{:}15|NodeB\}$
   $Fa(O)[idO35]$: $\{O_1 = 0.62|1|1|Today\ 12{:}30{:}15|NodeC\}$
   $Fa(O)[idO36]$: $\{O_1 = 0.60|1|1|Today\ 12{:}30{:}15|NodeD\}$

10. Not pattern recognition actions are declared.

11. The following facts about predictions for the next observations are calculated:

   $Fa(Ft)[idF5]$: $\{Ft_1 = 0.60|1|1|Today\ 12{:}30{:}15|NodeA\}$
   $Fa(Ft)[idF6]$:$\{Ft_1 = 0.75|1|1|Today\ 12{:}30{:}15|NodeB\}$
   $Fa(Ft)[idF7]$:$\{Ft_1 = 0.72|1|1|Today\ 12{:}30{:}15|NodeC\}$
   $Fa(Ft)[idF8]$: $\{Ft_1 = 0.62|1|1|Today\ 12{:}30{:}15|NodeD\}$

12. New facts related with adaptive thresholds are calculated:

   $Fa(Ath)[idA5]$: $\{Ath_1 = 0.61|1|1|Today\ 12{:}30{:}15|NodeA\}$
   $Fa(Ath)[idA6]$:$\{Ath_1 = 0.76|1|1|Today\ 12{:}30{:}15|NodeB\}$
   $Fa(Ath)[idA7]$:$\{Ath_1 = 0.73|1|1|Today\ 12{:}30{:}15|NodeC\}$
   $Fa(Ath)[idA8]\{Ath_1 = 0.63|1|1|Today\ 12{:}30{:}15|NodeD\}$

13. The rule-based inference engine processes the discovered facts by applying the rule Ru_1. Given that the condition $Fa(O_1) \geqslant Fa(AT_{h1})$ is satisfied for the data gathered by *NodeB*, The following fact related with $Self - QoSOverwatch$ is inferred.

   $Fa[idF1]$:$\{Fa(idO34) \geqslant Fa(idA2)|1|1|Today\ 12{:}30{:}15|NodeB\}$
   . Which describes the conclusion C_1:

   $C_1[idC1]$:$\{Suspicious\ QoS\ variation|SQoS|Fa(idO34) \geqslant Fa(idA2)\}$

14. The following symptom is reported to the Diagnosis layer:

   $Re_1[idR1]$:
   $\{Suspicious\ QoS\ variation|SQoS|idF1|1|idF1, idA2,\ idO34,\ Ru_1\}$

8 UC 3: Botnet detection

8.1 Description

The use case called *Self—BotnetMitigation* (SZombie) report symptoms related with Suspicious Command and Control (C&C) communications (Wang et al. 2017). In this case concurrency at pipelining is not applied, pattern recognition actions are considered, but prediction and adaptive thresholding are not required. Note that the external repositories (Rep1, Rep2) provide collections of legitimate (Rep1) and malicious (Rep2) traffic pattern observations on SELFNET. When a new discovered patter seems much more significantly to the collection of malicious patterns than the legitimate, a new fact that indicates this suspicious feature is acquired. Table 7 shows its onboarding descriptors according with the specification summarized in Table 2.

8.1.1 Step-by-step

The following illustrates and example of runtime in *Self—BotnetMitigation*, where different ADBs are loaded and analyzed according to the aforementioned indications. Figure 14 displays every step in a sequence diagram.

1. The descriptors of the use case *Self—BotnetMitigation* are loaded by the SELFNET Analyzer. Then, the memory for storing temporal containers of objects O_1, pattern recognition PR and facts Fa is allocated. The accessibility of the declared datasets D is verified.

2. The ADB_1 with aggregated instances of O_1 is requested to the Aggregation layer and then it is processed. The SELFNET Analyzer built the following facts from the information gathered by the network elements (*NodeA*, *NodeB*, *NodeC*, *NodeD*):

   $Fa\left(O_1\right)[idO1]$: $\{O_1 = FF217|1|1|Today\ 12{:}22{:}17|NodeA\}$
   $Fa\left(O_1\right)[idO2]$: $\{O_1 = 00DE8|1|1|Today\ 12{:}22{:}17|NodeB\}$
   $Fa\left(O_1\right)[idO3]$: $\{O_1 = F00FF|1|1|Today\ 12{:}22{:}17|NodeC\}$
   $Fa\left(O_1\right)[idO4]$: $\{O_1 = A4F09|1|1|Today\ 12{:}22{:}17|NodeD\}$

3. The two pattern recognition actions declared are executed. First, every $Fa\left(O_1\right)$ is correlated with the data-

Table 7 Self—BotnetMitigation specification

Item	Descriptor
Object	O_1:{*pattern*|1|1|*hexadecimal*}
Object	O_2:{*simLegi*|1|1|{0..1}}
Object	O_3:{*simMal*|1|1|{0..1}}
Operator	Op_1:{*Equal*| = | 1 |(*Fa*, *O*, *Va*) = (*Fa*, *O*, *Va*)|*equal*}
Operator	Op_2:{*LT*| > | 1 |(*Fa*, *O*, *Va*) > (*Fa*, *O*, *Va*)|*left is G*}
Dataset	D_{legi}:{*legitimatePatern*|*O*(*pattern*)|*collection*|*Rep*1}
Dataset	D_{mal}:{*maliciousPatern*|*O*(*pattern*)|*collection*|*Rep*2}
Pattern recognition	PR_1:{*legMeasure*|O_1|O_2|*anomaly*|$D(D_{legi})$}
Pattern recognition	PR_2:{*malMeasure*|O_1|O_3|*anomaly*|$D(D_{mal})$}
Conclusion	C_1:{*malicious communication*|*SZombie*|$Fa(O_2) < Fa(O_3)$}
Rule	Ru_1:{$Fa(O_2) < Fa(O_3) \rightarrow Fa(C_1)$|1|*SZombie*}

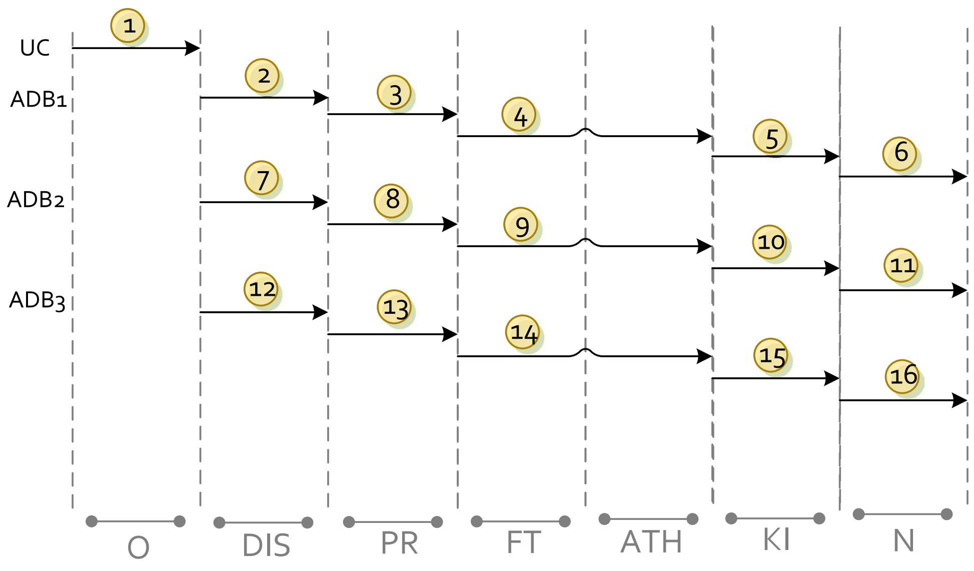


Fig. 14 Example of runtime in *Self—BotnetMitigation*

set D_{legi} looking for anomalies. The obtained anomaly scores O_2 allow acquiring the following facts:

$Fa(PR)[idL1]$: $\{O_2 = 0.9|1|1|Today\ 12{:}22{:}17|NodeA\}$
$Fa(PR)[idL2]$: $\{O_2 = 0.9|1|1|Today\ 12{:}22{:}17|NodeB\}$
$Fa(PR)[idL3]$: $\{O_2 = 0.8|1|1|Today\ 12{:}22{:}17|NodeC\}$
$Fa(PR)[idL4]$: $\{O_2 = 0.9|1|1|Today\ 12{:}22{:}17|NodeD\}$

On the other hand, every $Fa(O_1)$ is correlated with the dataset D_{mali} looking for anomalies. The obtained anomaly scores O_3 allow acquiring the following facts:

$Fa(PR)[idM1]$: $\{O_3 = 0.2|1|1|Today\ 12{:}22{:}17|NodeA\}$
$Fa(PR)[idM2]$: $\{O_3 = 0.1|1|1|Today\ 12{:}22{:}17|NodeB\}$
$Fa(PR)[idM3]$: $\{O_3 = 0.1|1|1|Today\ 12{:}22{:}17|NodeC\}$
$Fa(PR)[idM4]$: $\{O_3 = 0.1|1|1|Today\ 12{:}22{:}17|NodeD\}$

4. There are not predictions or adaptive thresholding actions to execute.
5. The rule-based inference engine processes the discovered facts by applying the rule Ru_1. Given that the condition $Fa(O_2) < Fa(O_3)$ is not satisfied, conclusions related with *Self—BotnetMitigation* are not inferred.
6. There are not symptoms to report. All the temporal facts are removed and the rule-based expert system is restarted.
7. The ADB_2 with aggregated instances of O_1 is requested to the Aggregation layer and then it is processed. The SELFNET Analyzer built the following facts from the information gathered by the network elements (*NodeA*, *NodeB*, *NodeC*, *NodeD*):

$Fa(O_1)[idO5]$: $\{O_1 = F2217|1|1|Today\ 12{:}23{:}17|NodeA\}$
$Fa(O_1)[idO6]$: $\{O_1 = 012E8|1|1|Today\ 12{:}23{:}17|NodeB\}$
$Fa(O_1)[idO7]$: $\{O_1 = F0211|1|1|Today\ 12{:}23{:}17|NodeC\}$
$Fa(O_1)[idO8]$: $\{O_1 = A2F18|1|1|Today\ 12{:}23{:}17|NodeD\}$

8. The two pattern recognition actions declared are executed. The following facts related with O_2 and O_3 are acquired:

$Fa(PR)[idL5]$: $\{O_2 = 0.5|1|1|Today\ 12{:}23{:}17|NodeA\}$
$Fa(PR)[idL6]$: $\{O_2 = 0.8|1|1|Today\ 12{:}23{:}17|NodeB\}$
$Fa(PR)[idL7]$: $\{O_2 = 0.7|1|1|Today\ 12{:}23{:}17|NodeC\}$
$Fa(PR)[idL8]$: $\{O_2 = 0.9|1|1|Today\ 12{:}23{:}17|NodeD\}$

$Fa(PR)[idM5]$: $\{O_3 = 0.6|1|1|Today\ 12{:}23{:}17|NodeA\}$
$Fa(PR)[idM7]$: $\{O_3 = 0.2|1|1|Today\ 12{:}23{:}17|NodeB\}$
$Fa(PR)[idM8]$: $\{O_3 = 0.2|1|1|Today\ 12{:}23{:}17|NodeC\}$
$Fa(PR)[idM9]$: $\{O_3 = 0.3|1|1|Today\ 12{:}23{:}17|NodeD\}$

9. There are not predictions or adaptive thresholding actions toexecute.
10. The rule-basedinference engine processes the discovered facts by applying the rule Ru_1. Given that the condition $Fa(O_2) < Fa(O_3)$is satisfied for the data gathered by *NodeA*, conclusions related with *Self—BotnetMitigation* are inferred. The following fact related with *Self—BotnetMitigation* is included to the working memory.

$Fa[idF1]$: $\{Fa(idL5) < Fa(idM5)|1|1|Today\ 12{:}23{:}17|NodeA\}$.
Which describes the conclusion C_1: $C_1[idC1]$: $\{malicious\ communication|SZombie|Fa(idL5) < Fa(idM5)\}$
11. The following symptom is reported to the Diagnosis layer:

$$Re_1[idR1]: \{mmalicious\ communication|SZombie|\ idF1|1|idF1, idL5,\ idM5,\ Ru_1\}$$
12. The ADB_3 with aggregated instances of O_1 is requested to the Aggregation layer and then it is processed. The SELFNET Analyzer built the following facts from the information gathered by the network elements (*NodeA*, *NodeB*, *NodeC*, *NodeD*):

$Fa(O_1)[idO9]$: $\{O_1 = F1110|1|1|Today\ 12{:}24{:}17|NodeA\}$
$Fa(O_1)[idOA]$: $\{O_1 = F2E80|1|1|Today\ 12{:}24{:}17|NodeB\}$
$Fa(O_1)[idOB]$: $\{O_1 = 11310|1|1|Today\ 12{:}24{:}17|NodeC\}$
$Fa(O_1)[idOC]$: $\{O_1 = AF42C|1|1|Today\ 12{:}24{:}17|NodeD\}$
13. The two pattern recognition actions declared are executed. The following facts related with O_2 and O_3 are acquired:

$Fa(PR)[idL9]$: $\{O_2 = 0.4|1|1|Today\ 12{:}24{:}17|NodeA\}$
$Fa(PR)[idLA]$: $\{O_2 = 0.3|1|1|Today\ 12{:}24{:}17|NodeB\}$
$Fa(PR)[idLB]$: $\{O_2 = 0.2|1|1|Today\ 12{:}24{:}17|NodeC\}$
$Fa(PR)[idLC]$: $\{O_2 = 0.9|1|1|Today\ 12{:}24{:}17|NodeD\}$

$Fa(PR)[idM9]$: $\{O_3 = 0.6|1|1|Today\ 12{:}24{:}17|NodeA\}$
$Fa(PR)[idMA]$: $\{O_3 = 0.5|1|1|Today\ 12{:}24{:}17|NodeB\}$
$Fa(PR)[idMB]$: $\{O_3 = 0.4|1|1|Today\ 12{:}24{:}17|NodeC\}$
$Fa(PR)[idMC]$: $\{O_3 = 0.2|1|1|Today\ 12{:}24{:}17|NodeD\}$

14. There are not predictions or adaptive thresholding actions to execute.
15. The rule-based inference engine processes the discovered facts by applying the rule Ru_1. Given that the condition $Fa(O_2) < Fa(O_3)$is satisfied for the data gathered by *NodeA*, *NodeB*and *NodeC*, conclusions related with *Self—BotnetMitigation* are inferred. The following facts related with *Self—BotnetMitigation* are included to the working memory.

$Fa[idF2]$: $\{Fa(idL9) < Fa(idM9)|1|1|Today\ 12{:}24{:}17|NodeA\}$

$Fa[idF3]$: $\{Fa(idLA) < Fa(idMA)|1|1|Today\ 12{:}24{:}17|NodeB\}$

$Fa[idF4]$: $\{Fa(idLB) < Fa(idMB)|1|1|Today\ 12{:}24{:}17|NodeC\}$
Which describes the conclusion
C_1:$C_1[idC2]$:$\{malicious\ communication|SZombie|$
$Fa(idL9) < Fa(idM9)\}$

$C_1[idC3]$:$\{malicious\ communication|SZombie|$
$Fa(idLA) < Fa(idMA)\}$

$C_1[idC4]$:$\{malicious\ communication|SZombie|$
$Fa(idLB) < Fa(idMB)\}$
16. The following symptoms are reported to the Diagnosis layer:

$$Re_2[idR2]: \{malicious\ communication|SZombie|\ idF2|1|idF2, idL9,\ idM9,\ Ru_1\}$$

$$Re_3[idR3]: \{malicious\ communication|SZombie|\ idF3|1|idF3, idLA,\ idMA,\ Ru_1\}$$

$$Re_4[idR4]: \{malicious\ communication|SZombie|\ idF4|1|idF4, idLB,\ idMB,\ Ru_1\}$$

9 Conclusions

This paper described the key elements of the SELFNET Analyzer Orchestrator, including the initial assumptions, design principles, workflows, sets of actions, execution strategies, and several examples of their application. These were properly deployed in the SELFNET project, where their effectiveness, configurability and extensibility were verified at different use cases (in particular, when applied to self-protection, self-optimization and self-healing capabilities). But even though our approach has proved to meet its design objectives, throughout the document has remained several aspects without covering, which depend directly on the implementation. For instance, some alternatives have been described for their optimization, but at the moment, only the basic task sequencing approach was being implemented on real uses cases. It brings support to the most basic requirements of the system, and allows the verification of the analyzed communication channels. But it is clear that there are many other ways to exploit parallelism at thread level, and therefore, to take more advantage of the analytic pipelining. On the other hand, the proposed scheme forces the most complex sets of actions to be executed in a certain way: pattern recognition, prediction, adaptive threshold construction and knowledge inference. But it is possible that future uses cases demand variations in their order; for example, that pattern recognition is being carried considering facts related with prediction and adaptive thresholds, or that once the knowledge inference tasks are completed, an additional step of predictions is required. At the moment, easy modifications on the SELFNET use case descriptors are able to overcome this inconvenience, but it is obvious that deepen in this problem is one of the main tasks of future work. Another aspect of interest is identifying quality indicators related with the granularity of the information contained in the ADBs. From them it is possible to improve the effectiveness of the analytic actions.

Acknowledgements This work is supported by the European Commission Horizon 2020 Programme under grant agreement number H2020-ICT-2014-2/671672 - SELFNET (Framework for Self-Organized Network Management in Virtualized and Software Defined Networks). Lorena Isabel Barona López is supported by the Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación SENESCYT (Quito, Ecuador) under Convocatoria Abierta 2013 Scholarship Program.

References

Baldo N, Giupponi L, Mangues-Bafalluy J (2014) Big data empowered self organized networks. In: Proceedings of 20th European wireless conference, Barcelona, pp 1–8

Barona López LI, Valdivieso Caraguay AL, Sotelo Monge MA, García Villalba LJ (2016) Key technologies in the context of future networks: operational and management requirements. Future Internet 9(1):1. doi:10.3390/fi9010001

Barona López LI, Valdivieso Caraguay AL, Maestre Vidal J, Sotelo Monge MA, García Villalba LJ (2017a) Towards incidence management in 5G based on situational awareness. Future Internet 9(1):3. doi:10.3390/fi9010003

Barona López LI, Maestre Vidal J, García Villalba LJ (2017b) An approach to data analysis in 5G networks. MDPI Entropy 9(2):1–23. doi:10.3390/e19020074

Bassiliades N, Vlahavas I, (1997) Processing production rules in DEVICE, an active knowledge base system. Data Knowl Eng 24(2):117–155. doi:10.1016/S0169-023X(97)00006-2

CHARISMA Project (2014) Converged heterogeneous advanced 5G cloud-RAN architecture for intelligent and secure media access. Funded under H2020-ICT-2014-2. Project Reference 671704. http://www.charisma5g.eu/. Accessed 11 Apr 2017

CROWD Project (2013) Connectivity management for energy optimised wireless dense networks. Funded under FP7-ICT. Project Reference 318115. http://www.ict-crowd.eu/. Accessed 11 Apr 2017

CVSS Forum of Incident Response and Security Teams (2015) CVSS: common vulnerability scoring system. https://www.first.org/cvss/specification-document. Accessed 11 Apr 2017

Endsley NR (1988) Design and evaluation for situation awareness enhancement. In: Proceedings of the human factors and ergonomics society annual meeting, Anaheim, 32(2):97–101

ENISA (2015) ENISA Threat Landscape 2015. https://www.enisa.europa.eu/publications/etl2015. Accessed 11 Apr 2017

Finkel A, Iyer SP, Sutre G (2003) Well-abstracted transition systems: application to FIFO automata. Inf Comput 181(1):1–31. doi:10.1016/S0890-5401(02)00027-5

5G-Ensure Project (2014) Enablers for network and system security and resilience. Funded under H2020-ICT-2014-2. Project Reference 671562. http://www.5gensure.eu/. Accessed 11 Apr 2017

5G-NORMA Project (2014) 5G NOvel radio multiservice adaptive network architecture. Funded under H2020-ICT-2014-2. Project Reference 671584. https://5gnorma.5g-ppp.eu/. Accessed 11 Apr 2017

5G-NOW Project (2013) 5th generation non-orthogonal waveforms for asynchronous signalling. Funded under FP7-ICT. Project Reference 318555. http://www.5gnow.eu/. Accessed 11 Apr 2017

Gordon MI, Thies W, Amarasinghe S (2006) Exploiting coarse-grained task, data, and pipeline parallelism in stream programs, In: Proceedings of the 12th international conference on architectural support for programming languages and operating systems, San Jose, pp 151–162

5G-PPP (2017) 5G infrastructure public private partnership. https://5g-ppp.eu. Accessed 11 Apr 2017

Guillaume S, Charnomordic B, (2012) Fuzzy inference systems: an integrated modeling environment for collaboration between expert knowledge and data using FisPro. Expert Syst Appl 39(10):8744–8755. doi:10.1016/j.eswa.2012.01.206

ISO International Organization for Standardization and the International Electrotechnical Commission (2005) ISO/IEC 27002: information technology, security techniques, code of practice for information security management. http://www.iso.org/iso/catalogue_detail?csnumber=54533. Accessed 11 Apr 2017

Leau YB, Ahmad A, Manickam S (2015) Network security situation prediction: a review and discussion. In: Proceedings of the 4th international conference on soft computing, intelligent systems, and information technology, Bali, pp 424–435

Lunardhi AD, Passino KM (1995) Verification of qualitative properties of rule-based expert systems. Int J Appl Artif Intell 9(6):587–621. doi:10.1080/08839519508945490

MCN Project (2013) Mobile cloud networking. Funded under FP7-ICT. Project Reference 318109. http://www.mobile-cloud-networking.eu/site/. Accessed 11 Apr 2017

METIS-II Project (2014) Mobile and wireless communications enablers for twenty-twenty (2020) information society-II. Funded under H2020-ICT-2014-2. Project Reference 671680. https://5g-ppp.eu/metis-ii/. Accessed 11 Apr 2017

Mijumbi R, Serrat J, Gorricho JL, Bouten N, Turck F, Boutaba R (2016) Network function virtualization: state-of-the-art and research challenges. IEEE Commun Surv Tuts 18(1):236–262. doi:10.1109/comst.2015.2477041

Neves P, Calé R et al (2016) The SELFNET approach for autonomic management in an NFV/SDN networking paradigm. Int J Distrib Sens Netw 12(2):1–17. doi:10.1155/2016/2897479

NIST National Institute of Standards and Technology (2007) NIST-SP800 series special publications on computer security. http://csrc.nist.gov/publications/PubsSPs.html#SP800. Accessed 11 Apr 2017

Osseiran A, Boccardi F et al (2014) Scenarios for 5G mobile and wireless communications: the vision of the METIS project. IEEE Commun Mag 52(5):26–35. doi:10.1109/mcom.2014.6815890

Qiao J, Shen XS, Mark JW, Shen Q, He Y, Lei L (2015) Enabling device-to-device communications in millimeter-wave 5G cellular networks. IEEE Commun Mag 53(1):209–215. doi:10.1109/MCOM.2015.7010536

SELFNET Project (2014) Framework for self-organized network management in virtualized and software defined networks. Funded under H2020-ICT-2014-2. Project Reference 671672. https://SELFNET-5g.eu/. Accessed 11 Apr 2017

SONATA Project (2014) Service programing and orchestration for virtualized software networks. Funded under H2020-ICT-2014-2. Project Reference 671517. http://www.sonata-nfv.eu/. Accessed 11 Apr 2017

Su J, Xu C, Chenung SC, Xi W, Jiang Y, Cao C, Ma X, Lu J (2016) Hybrid CPU–GPU constraint checking: Towards efficient context consistency. Inf Softw Tech 74:230–242. doi:10.1016/j.infsof.2015.10.003

T-NOVA Project (2013) Network functions as-a-service over virtualised infrastructures. Funded under FP7-ICT. Project Reference 619520. http://www.t-nova.eu/. Accessed 11 Apr 2017

UNIFY Project (2013) Unifying cloud and carrier networks. Funded under FP7-ICT. Project Reference 619609. http://www.fp7-unify.eu/. Accessed 11 Apr 2017

Wang YW, Hanson EN (1992) A performance comparison of the Rete and TREAT algorithms for testing database rule conditions. In: Proceedings of the 8th international conference on data engineering, Tempe, pp 88–97

Wang TS, Lin HT, Cheng WT, Chen CY (2017) DBod: clustering and detecting DGA-based botnets using DNS traffic analysis. Comput Secur 64:1–15. doi:10.1016/j.cose.2016.10.001

Webb J, Ahmad A, Maynard SB, Shanks G, Popovski P (2014) A situation awareness model for information security risk management. Comput Secur 44:1–15. doi:10.1016/j.cose.2014.04.005

Xia W, Wen Y, Foh CH, Niyato D, Xie H (2015) A survey on software-defined networking. IEEE Commun Surv Tuts. 17(1):27–51. doi:10.1109/comst.2014.2330903

Xu L, Assem H, Yahia IGB, Buda TS et al (2016) CogNet: a network management architecture featuring cognitive capabilities. In: Proceedings of the European conference on networks and communications, Athens, pp 325–329

Zou H, Yu Y, Tang W, Chen HWM (2014) FlexAnalytics: a flexible data analytics framework for big data applications with I/O performance improvement. Big Data Res 1:4–13. doi:10.1016/j.bdr.2014.07.001