Self-Aware Networks That Optimize Security, QoS, and Energy

In this article, self-awareness in networking is discussed. Examples of advantages of self-aware networks with respect to quality of service, energy, and security are presented.

By Erol Gelenbe, *Life Fellow IEEE*, Joanna Domanska, Piotr Fröhlich, Mateusz P. Nowak, and Sławomir Nowak

ABSTRACT | The need to adaptively manage computer systems and networks so as to offer good Quality of Service (QoS) and Quality of Experience (QoE) with secure operation at relatively low levels of energy consumption is challenged by their sheer complexity and the wide variability of the workloads. A possible way forward is through self-awareness, whereby self-measurement and self-observation, together with on-line control mechanisms, operate adaptively to attain the required performance and QoE. We survey the premises for these ideas arising from cognitive science and active networks and review recent work on self-aware computer systems and networks, including those that propose the use of software-defined networks as a means to implement these concepts. Then we provide some examples from the literature on self-aware systems to illustrate the performance gains that they can provide. Finally, we detail an example system and its working algorithms to allow the reader to understand how such a system may be implemented. Measurements showing how it can react rapidly to changing network conditions regarding QoS and security are presented. Some conclusions and suggestions for further work are listed.

KEYWORDS | Artificial intelligence (AI); cognitive packet network (CPN); quality of experience (QoE); quality of service (QoS); random neural networks (RNN); reinforcement learning (RL); self-aware networks; software-defined networks (SDNs).

Manuscript received May 30, 2019; revised November 8, 2019, April 1, 2020, and April 28, 2020; accepted April 29, 2020. Date of publication May 29, 2020; date of current version June 18, 2020. This research was supported by the European Commission under the H2020-IOT-2016-2017 Program, Grant Agreement 780139, for the SerIoT Research and Innovation Action. *(Corresponding author: Erol Gelenbe.)*

The authors are with the Institute of Theoretical and Applied Computer Science, Polish Academy of Sciences (IITiS-PAN), 44-100 Gliwice, Poland (e-mail: gelenbe.erol@gmail.com).

Digital Object Identifier 10.1109/JPROC.2020.2992559

I. INTRODUCTION

In the past few years, the telecommunications industry has woken up to the potential use of artificial intelligence (AI) and machine learning to automate and simplify network design, management, and operations. Thus, many organizations in the telecommunications industry have announced programs that aim at introducing machine learning into the next generation of packet and mobile networks. Recently, an industry publication [1] stated that "Advances in AI are pushing ... an economically feasible self-driving network ... that's programmed to ... carry out your intentions ... eliminates the complex programming and management tasks ... self-configure, monitor, manage, correct, defend ...with very little human intervention ... predict performance issues ... eliminate burdensome operational tasks and free ... IT staff ... costs will drop. Security, reliability, and resiliency will improve ... speed of business will accelerate."

Indeed, in the same line, a recent industry blog post [2] asks the rhetorical question "So how does AI help? It starts at the top, with codifying ... the intent of the network operator ... in human language or through a more traditional interface ... translated into network and security policies ... It is often especially important to use machine reasoning ... domain-specific knowledge about networking to ... realize the desired intent in the given network context ... with a deep understanding of the network infrastructure ... automates the policies ... optimizing for performance, reliability, and security."

In [3], mobile radio access networks (RANs), especially for 5G, are discussed, and it is suggested that AI can streamline RANs for massive multiple-input–multiple-output optimization with reinforcement learning (RL) [4] so that each cell self-adapts to changing scenarios and traffic, increasing throughput by 20% and optimizing speed for users that have low throughput.

Back in the 1990s, the research community had developed similar ideas [5]–[7] that have now met the technologies that allow their practical implementation. Thus, this article reviews the advances made since those early days in support of the current vision to use data analytics and machine learning to automate network operations through self-aware networks.

A. Premise From Cognitive Science and Philosophy

Self-awareness has been studied by cognitive scientists, Duval and Wicklund [8], who have discussed "conscious attention" as having two aspects: one being directed "toward the self," while the other is directed "toward the environment." This early conceptualization also emphasizes the importance of "discrepancies" between the internal model and the external reality. Discrepancies alert us on the differences between what we had thought about (from the internal model) and the reality we observe. Thus, discrepancies can be exploited via some form of RL because "the person will experience either negative or positive affect depending on whether attention is directed toward a negative or a positive discrepancy." This takes us back to Descartes' [9] rationalism and his well-known affirmation that "I think therefore I am," since thought leads to self-representation to the ability to link the internal model to external cues and events and improve the internal model.

On the other hand, similar insights into the notion of "self-awareness" can be gained through an interesting article by Brinck and Gärdenfors [10] that defines the intentionality of an agent which may be "cued," that is, triggered for instance by some sensory event or observation, or via a "detached" internal representation that models the world within which the agent exists. Links between the cued and detached representations are then based on experience. These links serve as a basis for action and for the refinement of the internal or detached model. This article argues about the importance of both the internal and detached representation that may include a self-model of the agent, which allows the agent to determine its capacity to have intentions and be able to act on these intentions. Both of these works by Duval and Wicklund [8] and Brinck and Gärdenfors [10] provide insights into the way in which many "self-aware" computer networks and computer systems have been designed with an internal model that is updated and corrected using observations based on measurement and RL to make corrections in the internal model and take decisions.

Self-awareness is also often discussed through the development of young animals and children [11], [12] who go through self-awareness that change with age. Another area that is often studied relates to anomalies in self-awareness. Examples include the commonly known "self-denial" which is viewed as a psychological protection mechanism whereby individuals refuse to acknowledge a weakness or an illness, and anosognosia [13] where a neurological disorder (i.e., malfunctioning of brain neural networks) can cause an individual's lack of awareness or denial of a well-identified physical incapacity or illness. Other work has considered the relation between internal representation of self-awareness and the capacity to take action [14].

B. Self-Aware Networks and Quality of Experience (QoE)

In the context of packet networks, self-awareness should include the ability of a network to pursue objectives or goal functions. It should be able to observe itself—via data from measurements that the network collects—and "criticize" its own behavior, so as to meet the objectives that have been set for it, and improve its behavior based on these observations. The network's primary goal should be to convey traffic flows from source to destination nodes and to retrieve and convey content to its users. However, this cannot be done without consideration for the QoE [15] of its "users," which are not just the end users that convey traffic or download content through the network but also the network's human operators and the people in charge of troubleshooting the network.

At the speed with which the network operates forwarding GBytes of data with hundreds of thousands of simultaneous users, the analysis of QoE does not mean that we can poll users about their satisfaction with the network [16] in real time, though some of that may happen at a slower pace by sampling some users. However, early work has successfully linked network bandwidth and perceived video quality, with the traffic characteristics and network quality of service (QoS) [17].

At the high-end, one may monitor the end-users' emotions as they use the network [18] to evaluate QoE, though the same emotions may be related to the content they receive rather than the network's own performance. However, substantial work has been conducted on linking the quality of the sound that the network users may be hearing through their connections to the QoS that is directly measured [19], [20]. A related strain of work has linked the perceived quality of video that is downloaded by the users to the usual traffic QoS measurements, such as packet loss and delay [21].

Another major element that enters into QoE, from the perspective of the network operator as a "user," is the network's energy consumption, because energy may represent close to 70% of a network's operating costs [22], [23]. Thus, the minimization of energy consumption [24], [25] needs to be included in the "goal" or objective of a self-aware network. Its measurement and evaluation is a key issue [26], [27] and it should also include the Cloud support that is indispensable for modern communication networks [28].

Resilience [29] and security [30] are also key issues for the network operator, who is the "user" that needs to assure the seamless nonstop operation of the system. Poor resilience and security are also directly felt by the end user. Thus, substantial research has been conducted in this area [31], while network software must be protected with appropriate recovery techniques [32]. Although the actions of the environment on a network arise mainly from the legitimate "good" users, "malicious" users can launch attacks or infiltrations which can have dramatic effects on the QoS perceived by end users. Thus, cyber security remains one of the core aspects of self-aware network management with the help of appropriate attack detection and mitigation techniques [33], [34].

Thus, in the sequel, when we express a self-aware network's goal or objective function, we will include both QoS, security, and energy, to handle the different issues related to QoE in a holistic manner.

C. Content of This Article

In Section II, we will consider the design of self-aware networks and discuss the first systematic attempt to incorporate intelligence in networks, known as active networks, which have provided many of the ideas that are still being pursued today.

Next, we will examine how self-aware networks can be built thanks to the rising technology of software-defined networks (SDNs), and how machine learning techniques such as RL can help implant self-awareness into such systems. This will lead to some detailed examples that will also quantitatively illustrate how they may be used in network overlay routing to reduce end-to-end packet delays in the global Internet or to reduce the average execution time of jobs that are allocated to edge or fog clusters, and in the Cloud.

Then we will go further into how SDN can incorporate self-awareness using the cognitive packet protocol based on smart packets (SPs) that gather measurements and information in the network and using an RL-based decision engine to modify the paths of flows dynamically to achieve better QoS.

This discussion will lead directly to a detailed presentation of the working example of a networked system whose aim is to provide QoS and energy aware network connectivity with active protection from network attacks. A test bed that embodies these concepts will be detailed, together with several measurements that illustrate its ability to react rapidly to QoS degradation and to security alarms.

This article ends with some conclusions and suggestions for further research.

II. RELATED WORK

The first mention of a "self-aware network" appears in 2003 [35], [36], while reference to an overlay network's "lower level network awareness" appears in [37], and context-awareness in wireless *ad hoc* networks is proposed in [38]. Another work has discussed "bodily self-awareness" with regard to humans and machines [14]. Self-aware services that can detect anomalies in Internet-based services are presented in [39] and web-aware tools are proposed in [40].

Yet the earlier concepts of cognitive radio (CR) [41] and cognitive packet networks (CPNs) from 1999 [7] also describe networks which are self-aware. Indeed, in CR [42], [43], the network's packet forwarders (in this case radio transmitters) directly sense the communication channel to enhance their awareness of the conditions under which they are communicating, before they transmit or forward packets. Their purpose is to optimize both the utilization of the channels, and the performance or QoS of the users' communications by reducing possible interference between distinct communications.

The work in [44] describes CPN [45] where "intelligent capabilities for routing and flow control are concentrated in the ... cognitive packets (which) route themselves. They are assigned goals ... and pursue these goals adaptively ... learn from their own observations about the network and from the experience of other packets with whom they exchange information via mailboxes." The analogy between a network path and a sequence of "letters" in the genetic code was also exploited in [46] to choose the best paths in CPN with a genetic algorithm that uses QoS as the fitness function.

Even earlier, the ALOHA network [47] was a pioneering initiative in self-aware "multiple access" networks to connect terminal consoles to computer servers via space satellite-based communications. Ideas from ALOHA were rapidly incorporated into the widely used local area network Ethernet [48], and also used in the first fiber-optics random access network Xantos [49], as well as the well-known space-satellite-based network Inmarsat for boats and ships [50].

ALOHA could be "slotted" so that time was synchronized throughout the network and all participants transmitted at the beginning of a slot, so that they could not be aware of each other prior to the transmission, but they could optimize the network's collisions by tracking the length of the silent periods to estimate the network's traffic rate [51], [52]. ALOHA could also be "unslotted" and operated in continuous time. In the latter case, the carrier sense multiple access (CSMA-CD) protocol [53] required the different users to sense the channel before transmitting, and refrain from transmitting if they "heard" that there were other ongoing transmissions on the channel, so as to minimize the probability of interference and collisions with other users [54].

A. Active Intelligent Networks

In the mid-1990s, the traditional role of Internet protocol (IP) networks that had emerged as the main follow-up to wired telecommunications were being seriously challenged. Indeed, IP networks were limited to transporting data passively and opaquely between systems, without taking advantage of the possibilities offered by the knowledge acquired about users and their requirements, and about the data content of packets. The capabilities offered by technological improvements that had increased the computational power of routers were not being exploited, and the routers' role was limited to managing packet headers and to signaling functions needed to manage connections.

This tradition was broken by the novel concept of active networking (AN) [6], [55] to perform useful tasks in a network to improve the end users' QoS and the network's performance, using software that is adaptively activated in the network. Examples include compressing high throughput packet flows (such as video) with knowledge of the data content, grouping the routing of multiple flows that carry identical media content into multicast trees to save network bandwidth and reducing end-to-end delay, using knowledge about a group of users in video-conferencing to optimize network topologies and minimize delay, or activating enhanced security and attack detection at nodes when they carry sensitive traffic.

AN places active "capsules" into IP packets that are either programs that can be run by routers, or data that activate or instantiate a program that is already resident at a node. It was suggested [56] that AN could enhance security by activating packet filters dynamically to authentify traffic flows. Capsules could also be used dynamically activate data caches in the network nodes. One major application of AN that was proposed was to deploy and update IP networks based on policies and specific network conditions [57], [58]. The worldwide interest it generated resulted in a global network Planetlab network of servers that survives till today [59].

Although AN initially encountered much enthusiasm, the idea did not fully survive and certain aspects were progressively incorporated into other technologies and research areas. AN had presented several difficulties, including the need to revisit the computing power and storage space available at network nodes, especially in those years when computing power was significantly lower than today. The unlimited capabilities of AN implied that the corresponding routers could find themselves overloaded with tasks, turning themselves into network servers with a role going way beyond routing, so that research had to be conducted into the design of more powerful network routers and corresponding test beds [60], [61]. In addition, AN could potentially create processing overload in the network nodes that came in addition to the load caused by the traffic itself, despite the fact that active routers could also interoperate with legacy routers, which transparently forward datagrams in a traditional manner [62].

Another major issue concerned security [63]. Indeed, the fact that AN proposed to use packets to inject programs into the network, with these programs being possibly specific to various users, could create tremendous security risks. Thus, each entering "caplet" would have to be monitored, either with regard to its provenance, with some form of admission control, or would have to be verified with some form of deep packet inspection. Both of these functions are not available in the IP, so that AN required a comprehensive review of IP. Thus, AN was raising even more questions regarding the means to mitigate the threats that it itself created, even though it could offer certain security services [64].

However, many good ideas that AN developed have been incorporated into current research. For instance, the cognitive or SPs in CPN [65] do not carry code but do carry data to instantiate programs that are already resident in the network nodes. The idea about caching data or creating services in routers or nodes has also been incorporated in many systems, and fog and edge computing carry some of the functions that were initially proposed for AN routers [66].

1) Self-Aware Networks That Use SDN: The rise of SDNs [67], [68] has provided a practical opportunity to experiment with networks that have some of the features that were suggested by AN. SDN can operate across various technologies including fiber-optic networks [69] and a combination of WiFi and wired connections [70].

In [71], an SDN controller incorporates machine learning-based decisions for routing so as to optimize QoS and tested to dynamically route flows between two fixed end servers in a small network with regard to its reactions to sudden changes of delay between nodes. A similar approach whose objective is to minimize energy was reported in [25]. However, the measurements show that an SDN controller operating with the CPN algorithm [65] switches paths rapidly and allows the test bed to reach its new performance level, within a few percent of the optimum end-to-end delay, in a few seconds. Such transients can be long in terms of the characteristic time constants of networks which operate at the millisecond level [72], and they should be studied further to determine their impact on the overall system optimization and the system's reaction delays.

There are several differences between the results in [71] and [73] which implement an RL-based SDN controller and measure its performance, and the interesting work by Lin *et al.* [74] presents numerical results concerning an RL algorithm's internal numerical values but not the system's resulting performance. Lin *et al.* [74] discussed a hierarchical structure, but then shows internal numerical values for the algorithm within a single controller so that the behavior of the system as a whole is not evaluated. The "single tree" in [71] can be used for one instance of a subtree in [74]. From an algorithmic perspective, in [74] only QoS is considered (and not security or energy), and it uses Markov decision processes with a soft-max decision rule, while in [73], a goal function is used, and direct system measurements are carried out on a test bed.

The research in [75] suggests the use of RL for load balancing in an IP network that is used to control the smart grid. It proposes to use an SDN controller for the network with two classes of data packets: those that carry monitoring traffic for the smart grid, and those that carry control commands which have a higher priority. A policy iteration approach based on Markov decision processes is proposed for the RL algorithm. The authors present discrete event simulations with Poisson traffic for a three node network to show the improvements that can be obtained regarding the network's QoS. Thus, although the work in this article presents an actual network implementation on a multiple node test bed with real traffic and experimental results both for QoS and security, the presentation in [75] covers QoS only, and it is illustrated with a discrete event simulation of three network nodes with idealized Poisson traffic.

Gelenbe *et al.* [76] and Nowak *et al.* [77] proposed an approach that combines SDN using CPN [65] and outlined a hierarchical vision of SDN similar to [74], without the algorithmic aspects and results of this article. In [78], a steady-state analysis of the radio transport layer for 5G is presented, based on prior mathematical work on signal to noise plus interference ratio calculations. An SDN configuration is then suggested to support base stations in order to maximize steady-state throughput at the radio level. This article differs from the problem we deal in this article which mainly focuses on an experimental investigation of wired networks and their dynamic real-time operation.

III. EXAMPLES OF SELF-AWARE SYSTEMS AND NETWORKS

The global Internet and the resulting possibility to create large-scale services with the help of the web, encouraged early work [79] on the design of systems that could dynamically monitor and improve QoS experienced by services for a large number of users, and examples were developed in web services for managing map data [40] and in detecting QoS and other anomalies in Internet services [39]. The latter example illustrates the inconsistency discussed earlier between what is expected based on an internal model, and the observations obtained from attention that is directed toward reality [8]. Early work [37] also focused on building network middleware without changes in the existing Internet substrate.

Another strain of work that has given impetus to the use of self-awareness in networks for various purposes is the field of "*ad hoc* networks" [80], [81] based on wireless nodes that are geographically distributed, yet relatively close to each other, and which dynamically create temporary links and network graph topologies with other nodes [38], [82]. Here, the nodes are often battery powered and mobile [83]. Since their energy consumption is of concern, if one wishes to maximize the failsafe operation of the network links [84], and neural network-based methods were implemented and tested to create self-aware methods to dynamically manage routes and maximize the energy life-time of the network [44].

A. Self-Aware Self-Managing Routing (SMART) Overlay Routing

In this section, we review results that were obtained when self-aware QoS oriented routing using RL was used with network overlays [85] motivated by the fact that many measurements have shown that the IP typically yields suboptimal network paths for QoS metrics [86].

The idea of overlay networks that use software installed in servers, or in machines that are connected to routers, so as to take routing decisions that supercede the ones taken by the routers themselves, has been around for some time [31]. One simple approach is to capture the packets coming from a given router, and then forward them to an IP address via the IP protocol that is not the "natural" next hop of the given packet, had it stayed within the given router. This decision can be taken based on prior knowledge that the modified next hop can provide better QoS for the packet toward the final destination. Various proposals and experiments are available in the literature regarding this idea [87], [88]. A simple approach would be to exploit "next hops" that are known to offer some QoS guarantee based on admission control schemes that were popular in earlier generation networks [89]. However, the approach taken here is based on self-awareness and online measurement.

Routing overlays have a long history and have also been suggested to improve the reliability and resilience of the network in the case of path outages or network attacks. However, they have the main advantage of overriding those routes that are selected by IP and route traffic based on the QoS considerations of the applications that are generating and utilizing the network connections. Potentially, many different applications may use overlays differently with different QoS considerations. The resilient overlay network (RON) [31] was the first such overlay for wide-area networks that demonstrated this advantage. However, RON had the disadvantage of having to monitor $O(M^2)$ connections for M overlay nodes and is thus was limited by the amount of overhead it imposed on each overlay node.

The SMART overlay overcomes this difficulty for an M node overlay network by limiting to four or another relatively small number of overlay neighbor nodes that a single overlay node can use as a next hop. Because of the use of self-awareness gained through the use of SPs and the RL algorithm described previously, SMART has demonstrated substantial improvements over IP in terms of both delay and throughput, while being very scalable.

The software overlay used by SMART is shown in Fig. 2, where we show various neighboring overlay nodes that are communicating with each other. The transmission agent intercepts packets entering the node and forwards

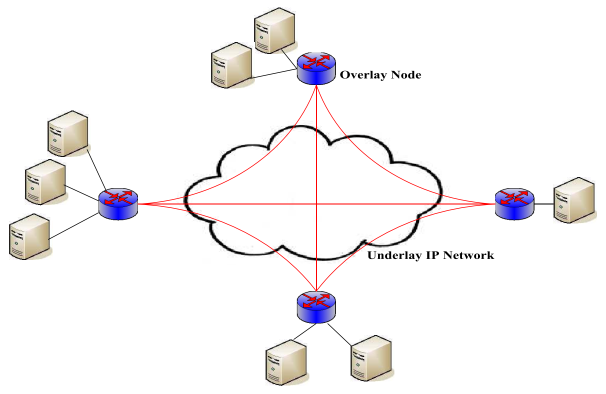
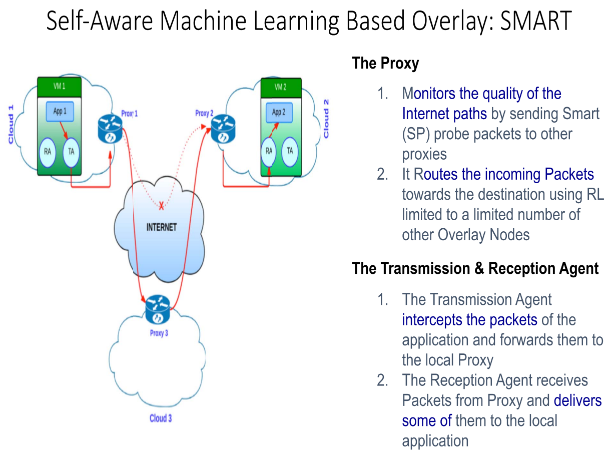


Fig. 1. ***Schematic representation of an overlay network.***

them to the proxy to be forwarded according to the SMART algorithm. The reception agent receives packets from the proxy, and if they are to be received locally, they are forwarded to the local application. Otherwise, the proxy forwards them one more hop according to the SMART algorithm. The proxy generates SPs that are used to monitor the network paths and forwards them, as well as other passing SPs toward the relevant neighboring node proxy. The proxy also operates the random neural network (RNN)-based [90] RL algorithm, with the data that are brought by the SPs.

SMART's data-driven intercontinental packet routing scheme regularly over time, say every 2 min, collects round-trip delay data between each overlay node and the other nodes to which it forwards packets. Each overlay node updates the RL algorithm's state and then updates the state of corresponding RNN [91] whose number of neurons is limited to the number of allowable neighboring overlay nodes. When it needs to send a packet to a given destination, the PROXY computes the activation probabilities of the corresponding RNN and selects the next neighboring overlay node to be used by the packet based on the neuron with highest probability, as described in Section III-C.



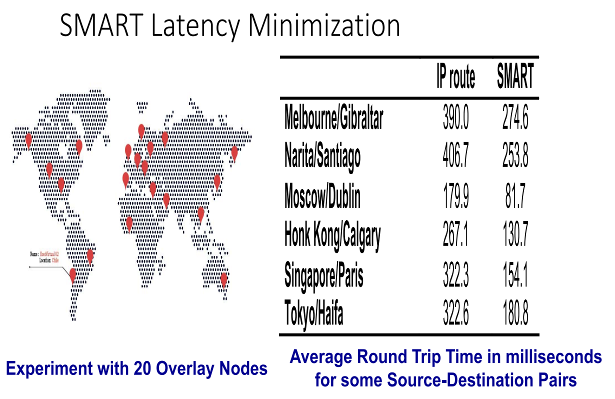
The Proxy

1. Monitors the quality of the Internet paths by sending Smart (SP) probe packets to other proxies
2. It Routes the incoming Packets towards the destination using RL limited to a limited number of other Overlay Nodes

The Transmission & Reception Agent

1. The Transmission Agent intercepts the packets of the application and forwards them to the local Proxy
2. The Reception Agent receives Packets from Proxy and delivers some of them to the local application

Fig. 2. ***Software architecture of the SMART overlay node.***


SMART Latency Minimization
Experiment with 20 Overlay Nodes
Average Round Trip Time in milliseconds for some Source-Destination Pairs

	IP route	SMART
Melbourne/Gibraltar	390.0	274.6
Narita/Santiago	406.7	253.8
Moscow/Dublin	179.9	81.7
Honk Kong/Calgary	267.1	130.7
Singapore/Paris	322.3	154.1
Tokyo/Haifa	322.6	180.8

Fig. 3. ***Average round-trip delay measured during a one-week experiment that compared the use of the IP protocol with SMART for several intercontinentally distributed overlay nodes. The data show a distinct reduction of delay when SMART is used.***

Experiments that were run by installing 20 overlay nodes in a large network offered by the NLNOG Ring test bed [92] are reported in [85]. Measurement results obtained with a fairly long one week experiment are shown in Fig. 3, and they exhibit a substantial reduction of round-trip packet delay as compared to the IP protocol, showing the advantage of a measurement-based self-aware scheme over a standard commonly used approach. More results related to these experiments can be found in [85] where results are also reported using overlays on the Amazon Cloud [93]. They all confirm the superiority of the self-aware approach for improving both delay and throughput in the network.

B. Self-Aware Scheduling of Tasks in the Cloud

Static algorithms for task allocation [94], [95] have long been preferred due to their low overhead and simplicity. However, they are only suitable for stable environments and cannot easily adapt to dynamic changes in the Cloud [96]. Dynamic algorithms [97], which can sometimes be quite complex, use the applications' characteristics prior to, and at, runtime, but often result in overheads that also cause performance degradations. Thus, typically dynamic adaptive schemes are evaluated through simulations [98] rather than in practical experiments. An example of such a fairly complex, nature-inspired task assignment, and load balancing algorithm can be found in [99]. Complex scheduling schemes that also use multiple hosts or servers must often be avoided because of the synchronization issues that can result in significant slowdowns [100].

Thus, in this section, we turn to the exploration of how self-awareness can be applied to the design of adaptive schemes that exploit online measurement and take decisions with low computational overhead to assign tasks to Cloud services [101], [102].

The experimental results that are reviewed in this are based on the work in [103] and [104] that develops a

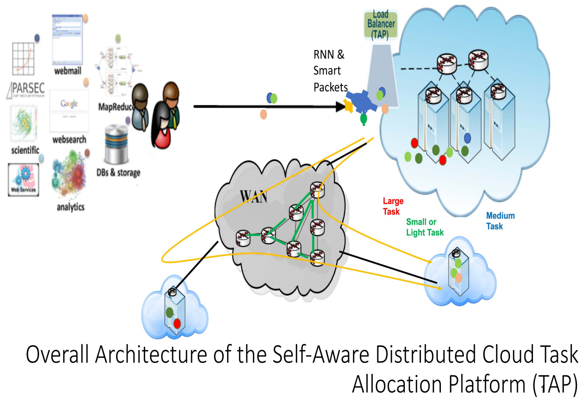


Fig. 4. ***Overall architecture and operation of the TAP which distributes tasks locally to a cloud server or distributes jobs remotely via the Internet to other fog and edge servers. On the tight-hand-side of the figure, we see that tasks may be of different types and may have wide ranging requirements in terms of memory occupancy and task execution time, while their QoS requirements may also vary widely.***

self-aware task allocation platform (TAP), implemented as a portable Linux-based system that dynamically allocates user tasks to available servers, exploiting online performance measurements of task and system performance. TAP attempts to meet the workload's service level agreements (SLAs), and it can support both static and dynamic allocation and load balancing schemes [105]. It collects measurements to provide performance reports and exploits these measurement results to take adaptive decisions.

TAP is used to allocate tasks to a collection of host servers, some of which may be distant from the others and remotely accessible through the Internet, while others may be collectively accessible as a Cloud server, as shown in Fig. 4. It runs on a host server and embeds measurement agents into each host server that it uses, either in a cloud, or at individual servers, to observe the system's state. The great variety of short, medium, and long tasks, and their different measured resource requirements are shown on the left-hand side of Fig. 5, while on the right-hand side of the same figure we show how different applications may have distinct service-level or QoS requirements that need to be respected, as discussed in [104].

These observations are then collected by SPs, as described in Section III-C. The SPs are forwarded by TAP at regular intervals to identify the subsystems which provide better performance, and each SP generates an acknowledgment (ACK) packet that comes back to TAP and carries the required measurement data. This provides TAP with a constant flow of information that TAP can use in its decision making.

The same TAP platform has been used to compare different task scheduling schemes, in order to see whether a self-aware approach would have distinct performance advantages. Although the work in [104] considers a larger set of task scheduling algorithms, including some that are based on a queuing analysis of the expected response times, here we will summarize results obtained with three schemes.

1) A round-robin allocation of incoming tasks to distinct hosts, so that irrespective of their size or of the host servers' workload, the tasks are shared equally among the servers. Round-robin provides a simple and practical baseline for comparison.
2) A "sensible" scheme [106] that allocates tasks to server i among a set of S servers, with a probability p_i that is inversely proportional to the measured average response time R_i, which includes the queuing and service delays at the server i
$$p_i = \frac{\frac{1}{R_i}}{\sum_{j=1}^{S} \frac{1}{R_j}}. \tag{1}$$
So that the system tends to load more those servers that provide better service. The assignments are randomized using a random number generator with the probability (1) for server i. Note that this algorithm will yield an average response time of
$$R = \sum_{j=1}^{S} p_i . R_i = \left[\sum_{j=1}^{S} \frac{1}{R_j} \right]^{-1}. \tag{2}$$
This algorithm also exploits the self-aware capabilities of TAP because it assigns load to a particular server as a function of all of the servers' ongoing measured performance.
3) Finally, a self-aware scheme that runs the RL algorithm in Section III-C, based on the observed task total execution times, including any wait time measured at the different servers.

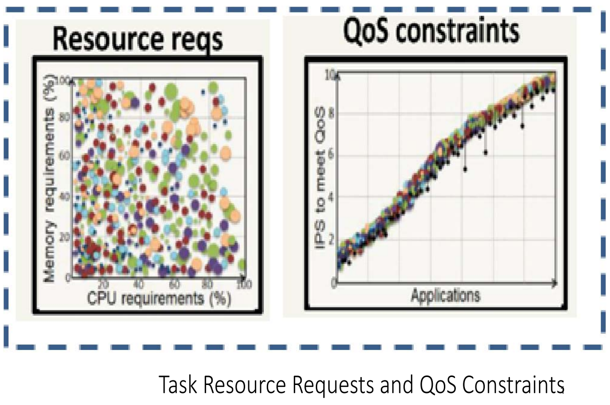


Fig. 5. ***Tasks may be of different types and their resource requirements may vary in terms of memory occupancy and task execution time. Their QoS requirements and SLAs may also vary widely.***

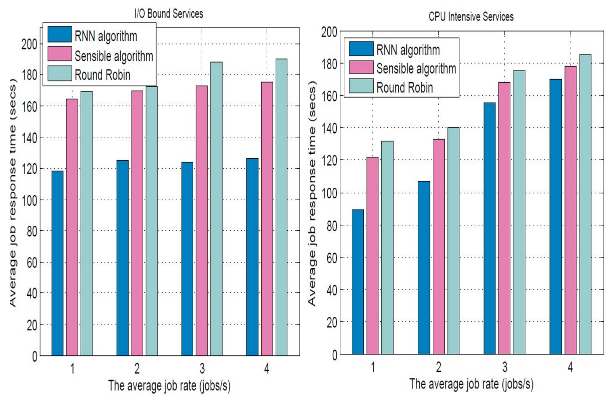


Fig. 6. ***Average response time for tasks that allocated to servers using RL and the RNN-based algorithm (marked RNN) as compared to the average response time observed by using round-robin scheduling and the sensible algorithm. The x-axis indicates increasing load levels in number of tasks arriving to TAP per unit time. For all load levels, the self-aware system that uses the CPN based on average response time provides the best performance. The "sensible" scheduler is the second best, while the round-robin heuristic offers the worst performance.***

This last approach differs from the "sensible" scheme in several ways. While the sensible scheme tends to spread the load over all of the servers but varies the fraction of tasks being assigned as a function of measured QoS, even if some of the servers provide poor performance, the last approach focuses at any time on a small set of the "best ones," and can stick to one "best" host server for some time, until that host server may become itself overloaded and provide poor performance. Furthermore, the use of (5) to update the RL scheme offers an in-built technique to encourage changes in decision making by "forgetfulness." This makes the self-aware scheme (c) more responsive to new data and more reactive with its ability to focus on the current small number of host servers that are able to provide the best performance.

Experimental data from [104] summarized in Fig. 6 shows that the RL-based task assignment scheme (c) results in significantly lower average response time (y-axis) for tasks at different levels of load, represented by the task arrival rates in the x-axis.

C. SDN With the CPN

The CPN protocol has been incorporated into SDN [71] for QoS-driven routing. It uses SPs which are sent out by each node e to the destination to measure the round-trip delay $D(f, e)$ from e to the destination node of the flow f. The packet loss rate $L(f, e)$ is measured by comparing the rate at which packets are forwarded for f and the rate at which ACK packets return. Similarly, at any node u of the network, it is possible to measure the power consumption in watts π_u, as well as the total traffic rate at the node λ_u, in packets/second, so that the average energy consumption per packet at the node is $E_u = (\pi_u/\lambda_u)$. These data are also collected by SPs.

$E(f, e)$, the total average energy consumed by a packet of flow f from node e until the destination, is then the sum of the energy consumed at each node from e to the destination.

The CPN approach uses a goal function, which describes the objective that the self-aware system is trying to minimize and whose value can be measured. At any node e, for any flow f traveling through it, the goal function $G(f, e)$ can include both the effect of delay $D(f, e)$, energy consumption $E(f, e)$, and packet loss rate $L(f, e)$

$$\begin{aligned} G(f,e) &= [b.D(f,e) + (1-b).E(f,e)][1-L(f,e)] \\ &\quad + L(f,e)[bD(f,e) + (1-b)E(f,e) \qquad (3) \\ &= \frac{bD(f,e) + (1-b)E(f,e)}{1-L(f,e)} \qquad (4) \end{aligned}$$

where $0 \leq b \leq 1$ and $(1 - b)$ provide a relative weight to the importance given to delay and energy. Thus, $G(f, e)$ is a composite goal function that the CPN tries to minimize as it decides how to forward packets.

From $G(f, e)$, the RL algorithm used by CPN computes the "reward" $R(f, e) = [G(f, e)]^{-1}$ at each node e using only locally available information that is collected by SPs and carried back by the ACK packet that corresponds to each SP. Each time such an ACK arrives at e a new value $R_l(f, e)$ becomes available at e, where l is the integer describing the successive values of the reward. The RL algorithm will first update the quantity

$$\theta_l = \alpha\theta_{l-1} + (1-\alpha)R_l, \quad 0 \leq \alpha < 1. \qquad (5)$$

So that θ_l describes the historical behavior of the reward, and tells how well the network has been doing, and a large value of θ_l represents "good" behavior.

The core decision element is the RNN [107], where each neuron corresponds to a distinct outgoing link of a router. Note that hardware implementations of the RNN have also been suggested [108], [109], and such additional hardware can potentially be installed in a routing engine.

For the case of task allocation discussed in Section III-B, each neuron corresponds to a distinct server that may be assigned to a task. The RNN has three useful properties.

1) It is a "recurrent" model, so that each neuron is interconnected with all other neurons, and allowing the RNN to represent the competition between neurons which recommend the choice of using different outgoing links of the network.
2) For a given numerical input and a given set of weights, the RNN state exists and is unique [110], that is, we are guaranteed to find a single solution to the state equations, which will be identical if the initial data are the same. This characteristic is very important for the reproducibility of the results and for the explainability of experimental outcomes.
3) The RNN state is easily computed from a nonlinear fixed-point computation.

The RL algorithm will then compute a set of RNN [110] connection weights as follows.

An N neuron RNN will be used, where N is the number of outgoing links for node e. With each outgoing link i from the node we associate the neuron i whose state is the "excitation probability" q_i. Let the RNN weights be the nonnegative real numbers $W^+_{ij}, W^-_{ij} \leq 0$ for $i, j \in \{1, \ldots, N\}$. From RNN theory [110], we know that

$$q_i = \frac{\lambda^+_i + \sum_{j=1}^N q_j W^+_{ji}}{r_i + \lambda^-_i + \sum_{j=1}^N q_j W^-_{ji}} \tag{6}$$

where $r_i = \sum_{j=1}^N [W^+_{ij} + W^-_{ij}]$ is the "total firing rate" of the neuron i, and λ^+_i, λ^-_i are, respectively, the arrival rate of excitatory and inhibitory spikes to neuron i from outside the neuron i. These rate parameters are set so that when all connection weights are of equal value, all the neurons in the RNN have an excitation probability of $q_i = 0.5$ representing an equal choice among all outgoing links.

The RNN's weights are updated as follows:

$$\begin{aligned}
&\text{If } R_l \geq T_{l-1}, \text{ then for } j \neq k\\
&\forall i \neq k, \quad W^+_{ik} \leftarrow W^+_{ik} + R_l,\ W^-_{ij} \leftarrow W^-_{ij} + \frac{R_l}{N-2}\\
&\text{If } R_l < T_{l-1}, \text{ then } j \neq k\\
&\forall i \neq k, \quad W^-_{ik} \leftarrow W^-_{ik} + R_l,\ W^+_{ij} \leftarrow W^+_{ij} + \frac{R_l}{N-2}
\end{aligned}$$

where the division by $N - 2$ is due to the fact that we are excluding the node I from which the SP initially arrived since we will not send the SP back and also not increasing the inhibitory weights of the winner nodes when $R_l \geq \theta_{l-1}$, nor will we increase the excitatory weights of the loser nodes when $R_l < \theta_{l-1}$.

Then we also renormalize the weights to avoid having weights that indefinitely increase or decrease

$$r^*_i \leftarrow \sum_{j=1}^N [W^+_{ij} + W^-_{ij}] \tag{7}$$

$$W^+_{ij} \leftarrow W^+_{ij} \frac{r_i}{r^*_i}, \quad W^-_{ij} \leftarrow W^-_{ij} \frac{r_i}{r^*_i}. \tag{8}$$

Finally, we calculate all the q_i from (6), select the new output link for flow f at node e by selecting the new output link k^*. Note that the node from which an SP entered the current node (where the next-hop decision is being taken) will not be used as the next hop since an SP is not allowed to head backwards along its path, so that the decision at a given node will only cover $N - 1$ other nodes. Thus, if I is the incoming link of a packet to be forwarded, we will choose the new outgoing link (or next hop) as being k^* as follows:

$$k^* = \text{argmax}\{q_i : i \neq I, 1 \leq i \leq N\}. \tag{9}$$

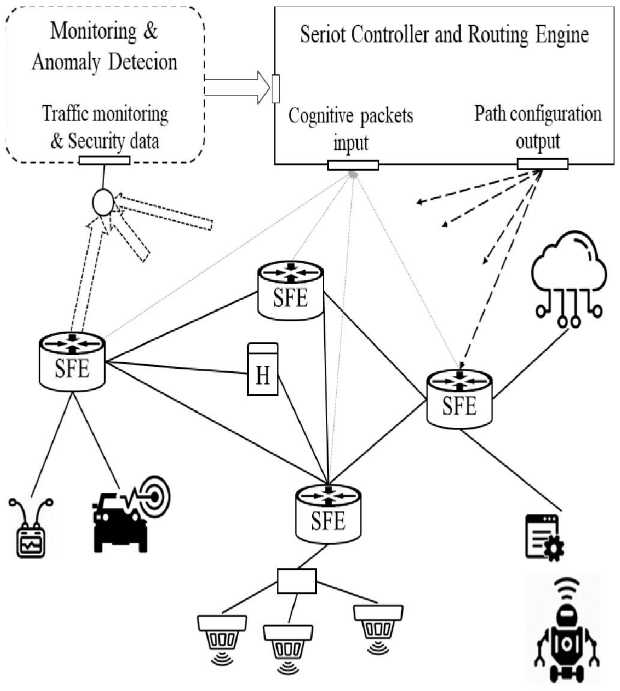

Monitoring & Anomaly Detection
Traffic monitoring & Security data
Seriot Controller and Routing Engine
Cognitive packets input
Path configuration output
SFE
SFE
H
SFE
SFE

Fig. 7. ***Overview of the networked architecture.***

IV. SELF-AWARE NETWORK FOR QOS, SECURITY, AND ENERGY CONSUMPTION

In this section, we describe the example of a practical working system that embodies many of the concepts and techniques that were described in Sections III-A–III-C, with the objective of showing how they can be used in a self-aware network, to optimize the QoE which was discussed in Section I-B, and combines in the same common a goal function three key components of QoE which are security, QoS, and energy consumption.

Many networks use low-end devices which may be battery operated. Some of them rely on sources of intermittent harvested energy. Thus, low energy consumption is a significant issue [111]. Furthermore some attacks, such as "denial of sleep" and battery attacks, directly aim at depleting rapidly the energy stored in battery-operated devices to disable the sensors and their networks [112], [113]. In addition, the overall energy consumption load due to networks, routers [114], and clouds [28] are also a critical issues.

A. System Architecture

The system architecture that we consider is shown in Fig. 7, where we show several smart forwarding elements (SFEs) or routers connected to each other. Each SFE can be connected to several fixed or mobile devices, or to gateways that themselves are connected to several devices (as at the bottom of the figure). Some of the SFEs will typically be connected to cloud servers (as at the top right-hand side of the figure) which may also consist of one or more fog servers. Some SFEs can be connected to an Internet of Things (IoT) or other device which comes under attack (as at the bottom right-hand side of the figure), and some SFEs may be connected to Honeypots (H in the middle of the figure) which collect and analyze attack data. Attacks are also detected by software at the SFEs and IoT gateways that carry out attack detection schemes similar to the work presented in [115].

The QoS, energy, and security data are being constantly collected throughout the network about the gateways and SFEs via SPs and is provided to the "Controller and routing engine" (top of the figure), operating as a standard OpenFlow SDN Controller to compute paths for the active, using the CPN algorithm. We call it the smart routing engine (SRE) whose is to update paths for the flows that are being carried in the network from source–destination pairs so as to minimize the goal function, and then communicate these paths to the SFEs using OpenFlow [116], [117].

Our SDN controller uses CPN routing [65] described in Section III-C, which is implemented with the RNN [107] and RL. Though such techniques appeared to be excessively advanced in the past, they are now attracting serious attention from the industry [118]. The SDN controller is an extension of a standard SDN network, whose principal components are as follows.

1) The SFEs, extending the concept of an SDN forwarder (or switch).
2) The smart controller(s) or routing elements built upon a standard SDN controller which is an open-source, professional grade SDN controller Open Network Operating System (ONOS) [67] is shown in Fig. 8.
3) Attack detectors [119]–[122] that are designed to detect attacks at edge devices and SFEs, and Honeypots [123] that are also installed at SFEs, or edge devices and servers, and used to attract potential attacks and to inform the controller by sending SPs to the SRE. The SPs will contain an attack detection probability that is included in the goal function that the SRE uses for routing control.

The heart of SRE shown in Fig. 8 is the cognitive routing module (CRM) which implements the decision making by RNNs [110], [124], responsible for path selection according to the current QoS, security, and energy consumption conditions in the network. The algorithm used by the CRM was described in Section III-C. The SPs, traveling from SFE to SFE, gather timestamps used for QoS evaluation, the current energy consumption at the SFEs (and possibly at edge devices when this information is available), and security data. The network state DB combines trust obtained from anomaly detection and forwarded in SPs to the SRE by SFEs, delays obtained via SPs, and energy per packet coming from energy consumption and traffic data also carried by SPs. The path translation (PT) module of the SRE translates the RNN decisions into

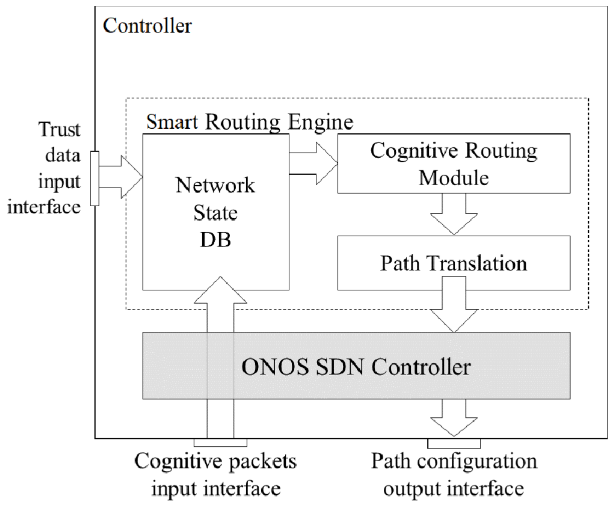


Fig. 8. ***SRE acting as an SDN controller that takes decisions for goal-based routing using the RNN and RL.***

path configuration commands, which are subsequently processed by the SRE and sent to specific SFEs using OpenFlow commands.

Each of SFE forwards the network's flows according to OpenFlow instructions that are sent to it by the SRE. SFEs are also able to collect data in the network with SPs which gather security, QoS, and energy usage data. On the other hand, SPs are routed by each SFE's internal cognitive packet agent (CPA). The agent unpacks the SP, adds its own data to the list stored inside it, packs it again, and forwards it to the next node based on the path set up by the SRE. After the SP has traveled a complete path, it carries the information given to it by all SFEs along the path. When a CPA recognizes that the SP it receives has reached the end of its path, it encapsulates the SP and sends it up to a specific data module, the Network State Database (NetStatDB) in the SRE. On the other hand, payload packets travel from SFE to adjacent SFE following a path that was set up by the SRE according to the SDN rules. Thus, SFEs handle both user packets and SPs.

Each SFE also sends standard network monitoring data (packet counters, byte counters) to the SRE, and an edge SFE will have client devices connected to it, such as IoT devices or edge servers.

B. Incorporating Security in the Goal Function

To show how new self-aware functionalities can be introduced into the system we have described, we illustrate the case of being able to recognize security issues and react to them in a timely manner. So, suppose that attack detectors placed at SFE's or at edge devices can generate security alerts that are then conveyed by SPs to the SRE. In that case, the SRE needs to be able to act upon these alerts. Thus, the goal function $G(f, e)$ must be extended and the expression in (3) should be modified to include security issues.

For some SFE or node e, and flow f, we define the Trust level $T(f,e)$, as a nonnegative number that says how much we can trust f when it flows through e, or reciprocally, how much f itself can trust e. This quantity can be provided either by the attack detector or by a Honeypot, or it can be based on some *a priori* knowledge concerning the flow f, for instance related to its source and destination nodes. Note that even when a flow is in one direction from source to destination, the source node can still be compromised by the flow via acknowledgment or other control packets that move back from intermediate or destination nodes toward the source or by other flows that pass enter that node.

Similarly, we define $S(f,e)$ the sensitivity of e when it is carrying f. This is a metric which says how tolerant the node e may be to events that are interpreted as a sign of insecurity in f, and $S(f,e)$ is again a nonnegative number.

With these concepts in mind, we have to indicate, how we take actions based on these metrics, and we define the insecurity factor $I(f,e)$ that "separates" e from f and vice versa

$$I(f,e) = \begin{cases} 0, & \text{if } S(f,e) \leq T(f,e) \\ S(f,e) - TF(f,e), & S(f,e) > T(f,e) \end{cases}$$

and using the notation $[X]^+ = X$ if $X > 0$, and $[X]^+ = 0$ if $X < 0$, we write

$$I(f,e) = 100 \cdot [S(f,e) - T(f,e)]^+ \qquad (10)$$

where the multiplicand 100 is a factor used to scale insecurity in comparison to the values of the QoS and energy consumption in the goal function. Clearly, if security is a critical issue this scaling factor will be set to a much larger value than the usually measured values of energy consumption per packet, and of delay per packet.

The parameters $S(f,e)$ and $T(f,e)$ can be easily set by the arrival to the SRE from a node e that is equipped with an attack detection mechanism such as the ones described in [122] and [125]. The SP will bring to the SRE the probability of an attack on node e, $P_A^N(e)$. Similarly, the flow attack probability $P_A^F(f)$ can be defined for some flow f via the probabilities that the nodes u which are on the path f, including the source and destination, have been attacked or compromised, via the expression

$$P_A^F(f) = 1 - \Pi_{u\in f}[1 - P_A^N(u)]. \qquad (11)$$

Let us illustrate the manner in which this may be used with two examples.

1) The flow f has a sensitivity to attacks of the order of 20%, that is, if the probability of an attack being detected is larger than 0.2, then the flow f feels uncomfortable about using e. In this case, we set $S(f,e) = 20$. Now suppose that the attack detector reports an attack probability $P_A^N(e) = 0.90$—then we set $T(f,e) = 90$, and this results in $I(f,e) = 100$.
2) The node e has a sensitivity to attacks of the order of 20%. Again, we set $S(f,e) = 20$, and if the attack detection system provides a probability $P_A^F(f) = 0.5$, then we obtain again $I(f,e) = 100$.

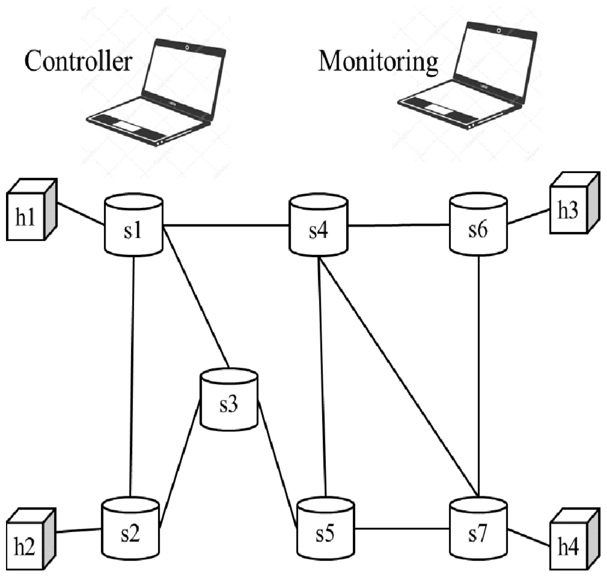


Fig. 9. ***Topology of the test bed.***

Thus, $S(f,e)$ acts as a threshold above which an attack becomes of concern. Note that the nonlinearity (10) can be modified to offer a more graduated, rather than step-function, response to security alarms. The goal function that will be used in our RL-based [126] routing scheme is extended from (3) to become

$$G(f,P) = \frac{bD(f,e) + cE(f,e) + (1-b-c)I(f,e)}{1 - L(f,e)} \qquad (12)$$

where $0 \leq b + c < 1$ are constants that allow us to weigh the relative importance of QoS, energy, and security.

C. Experimental Setting

To illustrate these ideas, laboratory test bed was set up with several SFEs that are implemented in lightweight Linux boxes with a quad-core ARMv8 processor running at 1 and 4 GHz, four Gigabit Ethernet interfaces and 2.4- and a 5-GHz 802.11 b/g/n/ac WiFi interface. SFEs were configured to use Ethernet ports as data plane interfaces, and WiFi as a management, monitoring, and SRE (controller) communication interface.

The data plane connections are represented schematically in Fig. 9 where

1) the symbols $s1, s2, \ldots, s7$ denote SFEs;
2) the symbols $h1, h2, h3$, and $h4$ denote terminal devices which are each emulated by 633-MHz MIPS

processor, a 100-Mb/s Ethernet port, and a 2.4-MHz WiFi connection that is used as a management port;

3) the controller (SRE) is installed on a separate workstation that is connected to the test bed via WiFi.

We have run many experiments and can emulate a variety of attacks, such as the effect of a worm which drops packets at random from the queue of a given SFE, the effect of distributed denial of service (DDoS) attacks, or the detection of an intrusion which would result in an increase of the numerical value of the trust metric $TF(.)$, indicating greater danger or mistrust, associated with a flow and a node. We can also represent QoS degradation by sudden increases in network traffic rates, or by introducing artificial delays or even complete stoppages (e.g., failures) of the SFEs.

However, due to space limitations, we focus on examples of the self-aware SDN's, that is, the SRE's, adaptive reactions to two types of effects: the degradation of QoS due to a significant increase in source to destination packet forwarding delay, and the degradation of security by an increase in the trust metric associated with a given path. The experiments we describe in this section are summarized in the real trace of events shown in Fig. 13.

The traffic in the network during the experiments was as follows.

1) Every distinct pair of clients from the set $\{h1, h2, h3, h4\}$ connected to the network generated the same flow of 20 packets per second each, that is, of the order of 20–40 kb/s. Thus, the network had 12 ongoing connections. Each individual flow's packet rate is compatible (and even quite high) with respect to IoT connections that may be monitoring physical conditions such as temperature of devices or rooms and water flow in pipes.
2) Additional traffic came from SPs generated by every edge node at ten packets per second, and the SRE's management traffic including OpenFlow commands, link discovery, topology discovery, and traffic statistics. We observed that the number of management packets passing through an SFE (router or forwarder), as observed using the Wireshark packet analyzer, was about four to five times higher than the SP traffic.
3) The measurements we present only concern one of the 12 end-user flows.

The experiments we report illustrate the ability of the network to adapt as a self-aware system. In particular,

1) we evaluate the reaction time of the network as a whole, to changes in the observed end-to-end delay of flows, so as to measure the network's reaction delays to sudden deterioration of QoS;
2) we measure the reaction time of the SRE itself to changes in the end-to-end delay of flows;
3) we track the changes to the important parameters of the RL algorithm: R_l and θ_{l-1} defined in (5), measured at successive steps $l = 1, 2, \ldots$;
4) we measure the reaction time of the SRE to changes in the security conditions represented by the level of trust;
5) we examine the behavior of the SRE in conditions that combine both the changes in path delay and the changes in the level of trust regarding the flows.

The SRE was programmed to update the network paths every 5 s. On the other hand, the measured SRE response times combine the delay related to the RL algorithm, the delay related to the RNN's computation, plus the network delay to receive data via SPs, together with the controller's own decision cycle. The resulting measurement does not give a clear picture of the speed at which the SRE works, but it does provide a realistic view of the overall reaction times perceived from the viewpoint of the network user.

D. System Reaction Times

As indicated in the literature [127], SDN routers introduce delays in decision making and network changes due to the need for the collection of data from the base routers, followed by the computation of a decision, followed in turn by the transfer of the decision to the base routers using the OpenFlow protocol. Thus, it was important in this article to actually measure the resulting effect when we use the self-aware routing algorithm.

We measured the packet round-trip time between hosts $h1$ and $h4$ of Fig. 9. The best path found by the controller was $\{s1, s4, s7\}$. After 5 s, the delay on link $s4 \rightarrow s7$ was changed in three distinct sets of experiments to the artificial values 100, 200, and 300 ms, and each trial was repeated independently 20 times.

We measured the time between the change in the value of the link delay (resulting immediately in an increase in the packet round-trip delay), and the installation of a new path, whose effect is observed by the return of the round trip delay to a lower value, since the new path does not use the link with the longer delay. The results are shown in Fig. 10.

Fig. 10 summarizes the resulting distribution of the network reaction time, measured with an accuracy of 1 s, over 20 trials for each of the color-coded delay values (100, 200, and 300 ms). Very interestingly, and as expected, we see that the reaction time is substantially better when the link delay is higher. Since we are considering the networked system as a whole, the reaction times are higher than those observed in Fig. 11.

1) SRE's Reaction Time to a Large Increase in Link Delay: The next experiments show the performance of the RNN-based algorithm in the routing engine (including the effect of the SPs which convey the QoS information), measured in a way that eliminates the impact of delays introduced by the SRE. The reaction time here is measured from the instant the link delay increases to the time when the RNN decision is made within the SRE but excludes the SDN time needed to change paths and inform the SFE. The relevant

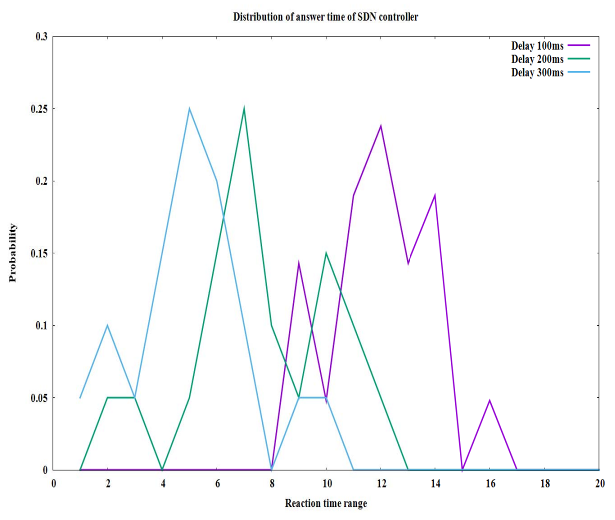


Fig. 10. ***Measured probability distribution of the reaction time of the system as a whole, including the network and the SRE, as viewed by the end user.***

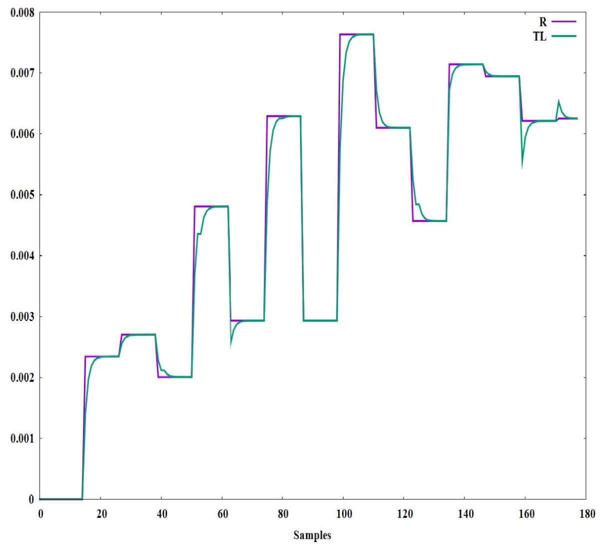


Fig. 12. ***Values of R_l and θ_l versus l the index of the sample.***

probability distributions, each resulting from 20 trials, and for the three values of link delay (100, 200, and 300 ms) are presented in Fig. 11. Again we see that the largest change in link delay (dotted green) results in the fastest reaction times, as would be expected. As expected, all times are shorter than those seen in Fig. 11 for the routing engine by itself.

E. Plotting the Values of the Reward

The values of R_l and θ_l are plotted versus the index l of successive events in Fig. 12. We see that that after each update of θ_l its value follows R_l, but the speed at which θ_l follows R_l will be affected by α in (5). In the present case, we used $\alpha = 0.4$. The instants when the two variables differ significantly is when the RNN-based algorithm "recommends" a path change to the SRE, which in turn will use the SDN protocol to forward the new paths to the SFEs.

Note also from Fig. 12 that in these measurements the system is starting from an empty state, hence the rise in the values of θ_l and R_l, but we see that the increase levels off at the right-hand side of the figure.

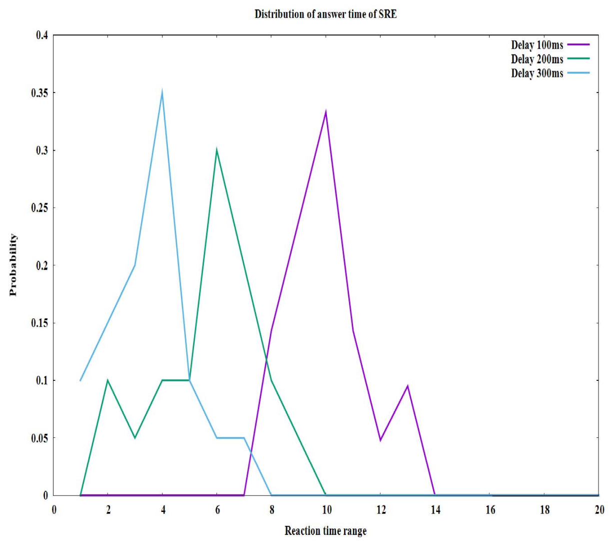


Fig. 11. ***Distribution of the routing engine's (SRE) reaction time, including the arrival of SPs to the SRE, and the effect of the RNN-based RL algorithm.***

F. Reaction to Changes in Network Delay and Trust Level

The impact of a change in the trust $T(.,.)$ level is shown in Fig. 13, where we present the probability distribution of the time it takes the SRE to respond to a large increase of 100 indicating some form of attack, in the value of $T(f,e)$ for a given e on the path being currently used. We note the roughly one second average delay showing the SRE's capacity to react very rapidly to attacks.

Finally, in Fig. 14, we show the observed delay across the network's response to changing delay (i.e., QoS in this case) and security conditions. At the first change point starting at the left of the figure, we see that when a very high delay of 300 ms appeared on the link, the system as a whole reacts in circa 2 s, finding a new path that does not use the deteriorated link. Then, as the $T(.,.)$ value rises significantly as an indication of the lack of security of the path that is being used, the network reroutes the traffic via a safe path, which includes a link with high delay. However, a small change in delay, results in a rapid path change to an insecure path with good QoS, and rapidly back. After circa

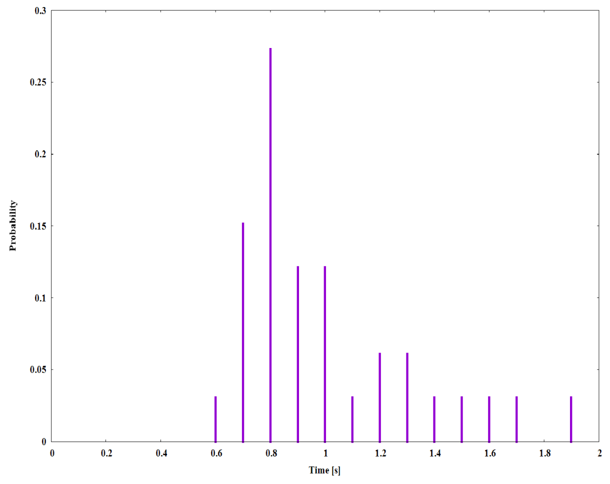


Fig. 13. ***Distribution of routing engine reaction time to a sudden substantial change in the trust metric T(., .).***

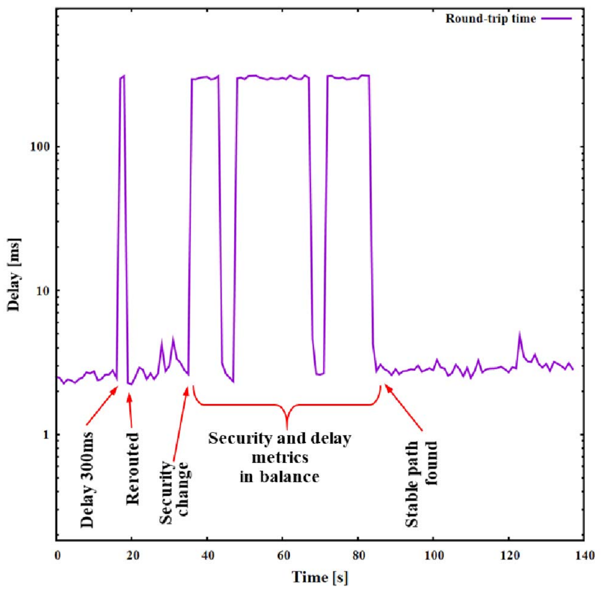


Fig. 14. ***System reactions to changes in delay and security. Note that the x-axis describes the course of real events in seconds, while the y-axis plots the reaction times of the system as a whole, in milliseconds, to the events that occur along the x-axis.***

50 s, the system manages to find a longer path (in number of nodes) which has a relatively low path delay and good security metric.

V. CONCLUSION AND FUTURE WORK

The widely distributed and highly interconnected structure of information processing systems, and the ever-changing nature of the underlying infrastructures makes it very difficult to control such systems in a top down hierarchical way. Thus, it is important to investigate means to manage and run such systems autonomously, based on self-measurement, local decisions, and self-optimization. The need for scalability imposes additional constraints, and the multiple of objectives of QoE, including QoS, energy savings, and system security offer further challenges. This took us to the concept of self-awareness, which allows a system to monitor itself and take decisions based on objectives that it pursues and observations regarding its own state.

Thus, in this article, we have started with a survey of approaches that incorporate self-awareness into networked systems. We have discussed the premises from cognitive science and also described early attempts related to self-observation in packet networks which resulted in widely used Ethernet-like systems. We have then discussed an early attempt to bring intelligence and adaptation into networks through the attractive concept of active networks and discussed some of the ideas that have then carried over into modern networking research.

Our focus has then turned to more recent work where self-awareness has been implemented in overlay networks and task management systems in the cloud for optimizing system performance. We have reviewed several proposals regarding the combination of recent advances in SDNs that create programmable controllers to dynamically manage paths in the network to achieve significant performance improvements. These discussions have been completed with experimental results that show the significant performance improvements that can be obtained when real-time self-aware adaptive management is implemented in an overlay network and in a cloud.

Then, we have presented an approach to introducing self-awareness into SDN through a CPN which has specific performance goals it pursues, and which is implemented through an RL algorithm that is incorporated into SDN controllers. It was illustrated then via a specific implementation in a multihop network test bed where the SDN controller aims at optimizing QoE including QoS, security, and energy. Experimental results have been shown concerning the responsiveness of the system, and its ability to react rapidly to sudden degradation in network delay or in security levels.

This broad panorama going from historical premises all the way to a system demonstrator is meant to entice the reader's curiosity and encourage the reader to investigate this area further. Many things remain yet to be done.

The integration of such techniques into hierarchical structures is a worthwhile direction of research. We have incorporated QoS, energy, and security together, and these areas are not distinct since they strongly interact: cyber attacks degrade QoS and increase energy consumption, while the optimization of QoS will itself increase energy consumption through additional computation and communication. Similarly, by reducing energy consumption we may also slow down the system as a whole. Thus, all these interactions are quite complex and will require further work.

Similarly, it will be useful to study the best possible machine learning techniques that may be used. For instance, dynamic system management based, not just on short-term observation through RL but also using long-term experience and big data, may be an alternative way to approach these interesting problems. Future work should also make use of predictions from performance models to improve and optimize the design of such networked systems [128], [129]. ■

Acknowledgment

The authors have benefited greatly from the recommendations of the referees and editors, to whom they are deeply indebted.

REFERENCES

[1] Juniper-Networks. (2020). *Expel Complexity With a Self-Driving Network: Soon, Your Network Will Adaptively Meet Your Business Goals all by Itself*. [Online]. Available: https://www.juniper.net/us/en/dm/the-self-driving-network/

[2] J. Apostolos. (Jun. 2019). *Improving Networks With Artificial Intelligence*. [Online]. Available: https://blogs.cisco.com/networking/improving-networks-with-ai

[3] R. Kompany. (Dec. 2018). *Huawei's 'Autonomous Driving' Mobile Networks Strategy Aims to Increase Automation and Reduce Costs, Knowledge Centre*. [Online]. Available: https://www.analysysmason.com/Research/Content/Comments/Huawei-autonomous-network-RMA18/

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning—An Introduction* (Adaptive Computation and Machine Learning). Cambridge, MA, USA: MIT Press, 1998. [Online]. Available: http://www.worldcat.org/oclc/37293240

[5] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. Adv. Neural Inf. Process. Syst. 7th NIPS Conf.*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds. Denver, CO, USA: Morgan Kaufmann, 1993, pp. 671–678.

[6] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *Comput. Commun. Rev.*, vol. 26, pp. 5–18, Oct. 1996.

[7] E. Gelenbe, Z. Xu, and E. Seref, "Cognitive packet networks," in *Proc. 11th Int. Conf. Tools Artif. Intell.*, Chicago, IL, USA, Nov. 1999, pp. 47–54, doi: 10.1109/TAI.1999.809765.

[8] S. Duval and R. A. Wicklund, *A Theory of Objective Self Awareness*. New York, NY, USA: Academic, 1972.

[9] R. Descartes, "Discours de la méthode," I. Maire, Printer, Leyde, The Netherlands, Tech. Rep., 1637.

[10] I. Brinck and P. Gärdenfors, "Representation and self-awareness in intentional agents," *Synthese*, vol. 118, no. 1, pp. 89–104, 1999, doi: 10.1023/A:1005109414345.

[11] K. A. Bard, B. K. Todd, C. Bernier, J. Love, and D. A. Leavens, "Self-awareness in human and chimpanzee infants: What is measured and what is meant by the mark and mirror test?" *Infancy*, vol. 9, no. 2, pp. 191–219, 2006.

[12] P. Rochat, "Five levels of self-awareness as they unfold early in life," *Consciousness Cognition*, vol. 12, no. 4, pp. 717–731, Dec. 2003.

[13] K. M. Heilman, A. M. Barrett, and J. C. Adair, "Possible mechanisms of anosognosia: A defect in self-awareness," *Phil. Trans. Roy. Soc. London B, Biol. Sci.*, vol. 353, no. 1377, pp. 1903–1909, Nov. 1998.

[14] J. L. Bermúdez, "Bodily self-awareness and the will: Reply to power," *Minds Mach.*, vol. 11, no. 1, pp. 139–142, 2001, doi: 10.1023/A:1011239215879.

[15] G. Harman, "The intrinsic quality of experience," *Phil. Perspect.*, vol. 4, Jan. 1990, pp. 31–52. [Online]. Available: http://www.jstor.org/stable/2214186

[16] N. Seufert *et al.*, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 469–492, 1st Quart., 2015.

[17] C. E. Cramer and E. Gelenbe, "Video quality and traffic QoS in learning-based subsampled and receiver-interpolated video sequences," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 2, pp. 150–167, Feb. 2000.

[18] C. Aracena, S. Basterrech, V. Snael, and J. Velasquez, "Neural networks for emotion recognition based on eye tracking data," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 2632–2637, doi: 10.1109/SMC.2015.460.

[19] S. Jelassi and G. Rubino, "A study of artificial speech quality assessors of VoIP calls subject to limited bursty packet losses," *EURASIP J. Image Video Process.*, vol. 2011, no. 1, p. 9, Dec. 2011.

[20] S. Jelassi and G. Rubino, "A perception-oriented Markov model of loss incidents observed over VoIP networks," *Comput. Commun.*, vol. 128, pp. 80–94, Sep. 2018, doi: 10.1016/j.comcom.2018.06.009.

[21] N. Torres-Cruz, M. E. Rivero-Angeles, G. Rubino, R. Menchaca-Mendez, and R. Menchaca-Mendez, "A window-based, server-assisted P2P network for VoD services with QoE guarantees," *Mobile Inf. Syst.*, vol. 2017, pp. 1–18, Mar. 2017, doi: 10.1155/2017/2084684.

[22] E. Gelenbe and Y. Caseau, "The impact of information technology on energy consumption and carbon emissions," *Ubiquity*, vol. 2015, pp. 1–15, Jun. 2015.

[23] C. Lange, D. Kosiankowski, R. Weidmann, and A. Gladisch, "Energy consumption of telecommunication networks and related improvement options," *IEEE J. Sel. Topics Quantum Electron.*, vol. 17, no. 2, pp. 285–295, Mar. 2011.

[24] E. Gelenbe and C. Morfopoulou, "A framework for energy-aware routing in packet networks," *Comput. J.*, vol. 54, no. 6, pp. 850–859, Jun. 2011.

[25] E. Gelenbe and T. Mahmoodi, "Energy-aware routing in the cognitive packet network," in *Proc. Energy, 1st Int. Conf. Smart Grids, Green Commun. IT Energyaware Technol.*, Venice, Italy, May 2011, pp. 7–12. [Online]. Available: https://www.semanticscholar.org/paper/Energy-Aware-Routing-inthe-Cognitive-Packet-Gelenbe/f01240825b8ffafe856e84549323337f62f15718

[26] E. Gelenbe, "Energy packet networks: Adaptive energy management for the cloud," in *Proc. 2nd Int. Workshop Cloud Comput. Platforms (CloudCP)*, 2012, p. 1.

[27] O. H. Abdelrahman and E. Gelenbe, "Time and energy in team-based search," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 87, no. 3, Mar. 2013, Art. no. 032125.

[28] A. Berl *et al.*, "Energy-efficient cloud computing," *Comput. J.*, vol. 53, no. 7, pp. 1045–1051, 2010.

[29] J. P. G. Sterbenz *et al.*, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Comput. Netw.*, vol. 54, no. 8, pp. 1245–1265, Jun. 2010.

[30] E. Gelenbe and G. Loukas, "A self-aware approach to denial of service defence," *Comput. Netw.*, vol. 51, no. 5, pp. 1299–1314, Apr. 2007.

[31] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. 18th ACM Symp. Oper. Syst. Princ. (SOSP)*, 2001, pp. 131–145.

[32] E. Gelenbe and S. Tucci, "Performances d'un système informatique dupliquè," *Comp. rendus de l'Académie des sciences. Série 2, Mécanique, Physique, Chimie, Sciences de l'univers, Sciences de la Terre*, vol. 312, no. 1, pp. 27–30, 1991.

[33] G. Oke, G. Loukas, and E. Gelenbe, "Detecting denial of service attacks with Bayesian classifiers and the random neural network," in *Proc. IEEE Int. Fuzzy Syst. Conf.*, Jun. 2007, pp. 1–6.

[34] R. Lent, G. Sakellari, and G. Loukas, "Strengthening the security of cognitive packet networks," *Int. J. Adv. Intell. Paradigms*, vol. 6, no. 1, pp. 14–27, 2014, doi: 10.1504/IJAIP.2014.059584.

[35] E. Gelenbe, "Review of experiments in self-aware networks," in *Proc. 18th Int. Symp., Comput. Inf. Sci. (ISCIS)* (Lecture Notes in Computer Science), vol. 2869, A. Yazici and C. Sener, Eds. Antalya, Turkey: Springer, Nov. 2003, pp. 1–8, doi: 10.1007/978-3-540-39737-3_1.

[36] E. Gelenbe and A. Núñez, "Self-aware networks and quality of service," in *Proc. Joint Int. Conf., Artif. Neural Netw. Neural Inf. Process. (ICANN/ICONIP)* (Lecture Notes in Computer Science), vol. 2714, O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, Eds. Istanbul, Turkey: Springer, Jun. 2003, pp. 901–908, 10.1007/3-540-44989-2_107.

[37] L. Massoulie, A.-M. Kermarrec, and A. J. Ganesh, "Network awareness and failure resilience in self-organizing overlay networks," in *Proc. 22nd Int. Symp. Reliable Distrib. Syst.*, Florence, Italy, Oct. 2003, pp. 47–55, doi: 10.1109/RELDIS.2003.1238054.

[38] K. Paul and D. Westhoff, "Context aware detection of selfish nodes in DSR based ad-hoc networks," in *Proc. IEEE 56th Veh. Technol. Conf.*, Taipei, Taiwan, Sep. 2002, pp. 2424–2429, doi: 10.1109/VETECF.2002.1040656.

[39] A. Bronstein *et al.*, "Self-aware services: Using Bayesian networks for detecting anomalies in Internet-based services," in *Proc. IEEE/IFIP Int. Symp. Integr. Netw. Manage. (IM)*, G. Pavlou, N. Anerousis, and A. Liotta, Eds., Seattle, WA, USA, May 2001, pp. 623–638, doi: 10.1109/INM.2001.918070.

[40] S. Li, "Leveraging a Web-aware self-organization map tool for clustering and visualization," in *Proc. 1st Asia–Pacific Conf., Web Intell., Res. Develop. (WI)* (Lecture Notes in Computer Science), vol. 2198, N. Zhong, Y. Yao, J. Liu, and S. Ohsuga, Eds. Maebashi City, Japan: Springer, Oct. 2001, pp. 579–583, doi: 10.1007/3-540-45490-X_75.

[41] J. Mitola and G. Q. Maguire, "Cognitive radio: Making software radios more personal," *IEEE Pers. Commun.*, vol. 6, no. 4, pp. 13–18, 1999.

[42] J. Mittola, "Cognitive radio—An integrated agent architecture for software defined radio," Ph.D. dissertation, KTH Royal Inst. Technol., Kista, Sweden, 1990.

[43] C. Stevenson, G. Chouinard, Z. Lei, W. Hu, S. Shellhammer, and W. Caldwell, "IEEE 802.22: The first cognitive radio wireless regional area network standard," *IEEE Commun. Mag.*, vol. 47, no. 1, pp. 130–138, Jan. 2009.

[44] E. Gelenbe, R. Lent, and Z. Xu, "Networks with cognitive packets," in *Proc. 8th Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst. (MASCOTS)*. San Francisco, CA, USA: IEEE Computer Society, Aug./Sep. 2000, pp. 3–10, doi: 10.1109/MASCOT.2000.876422.

[45] E. Gelenbe, "Cognitive packet network," U.S. Patent 09 680 184, Oct. 12, 2004.

[46] E. Gelenbe, P. Liu, and J. Lainé, "Genetic algorithms for route discovery," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 6, pp. 1247–1254, Dec. 2006.

[47] N. Abramson, "THE ALOHA SYSTEM: Another alternative for computer communications," in *Proc. Fall Joint Comput. Conf.* Las Vegas, NV, USA: AFIPS Press, 1970, pp. 281–285. [Online]. Available: https://people.eecs.berkeley.edu/~pister/290Q/Papers/MAC%20protocols/ALOHA%20abramson%201970.pdf

[48] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Commun. ACM*, vol. 26, no. 1, pp. 90–95, Jan. 1983.

[49] E. Gelenbe, J. Geyl, O. Gibergues, and E. Horlait, "Réseau local à diffusion de pacquets sur fibre optique: Le système Xantos," Rapport de Recherche LRI, Res. Rep., 1979, no. 63.

[50] D. Sagar, "The privatization of inmarsat," in *Proc. IISL 41st Colloq. Law Outer Space*, 1998, pp. 205–223.

[51] G. Fayolle, E. Gelenbe, and J. Labetoulle, "Stability and optimal control of the packet switching broadcast channel," *J. ACM*, vol. 24, no. 3, pp. 375–386, Jul. 1977.

[52] B. T. An and E. Gelenbe, "Near optimal behaviour of the packet switching broadcast channel," in *Proc. IEEE Comput. Soc., 2nd Int. Comput. Softw. Appl. Conf. (COMPSAC)*, Chicago, IL, USA, Nov. 1978, pp. 728–734, doi: 10.1109/CMPSAC.1978.810557.

[53] L. Kleinrock and F. Tobagi, "Packet switching in radio channels: Part I—Carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Trans. Commun.*, vol. COM-23, no. 12, pp. 1400–1416, Dec. 1975.

[54] E. Gelenbe and I. Mitrani, "Control policies in CSMA local area networks: Ethernet controls," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 11, no. 4, pp. 233–240, Dec. 1982.

[55] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 2, pp. 7–23, Apr. 1999.

[56] K. Psounis, "Active networks: Applications, security, safety, and architectures," *IEEE Commun. Surveys Tuts.*, vol. 2, no. 1, pp. 2–16, 1st Quart., 1999.

[57] A. Galis, S. Denazis, C. Brou, and C. Klein, *Programmable Networks for IP Service Deployment*. Norwood, MA, USA: Artech House, 2004.

[58] J. Rubio-Loyola *et al.*, "Platforms and software systems for an autonomic Internet," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2010, pp. 1–6.

[59] M. Hicks, J. T. Moore, D. S. Alexander, C. A. Gunter, and S. M. Nettles, "PLANet: An active Internet work," in *Proc. IEEE Conf. Comput. Commun., 18th Annu. Joint Conf. IEEE Comput. Commun. Societies Future (INFOCOM)*, New York, NY, USA, Mar. 1999, pp. 1124–1133, doi: 10.1109/INFCOM.1999.751668.

[60] D. S. Decasper, B. Plattner, G. M. Parulkar, S. Choi, J. D. DeHart, and T. Wolf, "A scalable high-performance active network node," *IEEE Netw.*, vol. 13, no. 1, pp. 8–19, Jan./Feb. 1999.

[61] A. Dollas *et al.*, "Architecture and application of Plato, a reconfigurable active network platform," in *Proc. 9th Annu. IEEE Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Mar. 2001, pp. 101–110.

[62] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Commun. Mag.*, vol. 35, no. 1, pp. 80–86, Jan. 1997.

[63] D. S. Alexander, W. A. Arbaugh, A. D. Keromytis, and J. M. Smith, "Security in active networks," in *Secure Internet Programming, Security Issues for Mobile and Distributed Objects* (Lecture Notes in Computer Science), vol. 1603, J. Vitek and C. D. Jensen, Eds. Berlin, Germany: Springer, 1999, pp. 433–451.

[64] D. S. Alexander, P. B. Menage, A. D. Keromytis, W. A. Arbaugh, K. G. Anagnostakis, and J. M. Smith, "The price of safety in an active network," *J. Commun. Netw.*, vol. 3, no. 1, pp. 4–18, Mar. 2001.

[65] E. Gelenbe, "Steps towards self-aware networks," *Commun. ACM*, vol. 52, no. 7, pp. 66–75, Jul. 2009.

[66] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, Aug. 2012, pp. 13–16.

[67] P. Berde *et al.*, "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw. (HotSDN)*. New York, NY, USA: ACM, 2014, pp. 1–6, doi: 10.1145/2620728.2620744.

[68] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *J. Netw. Comput. Appl.*, vol. 67, pp. 1–25, May 2016, doi: 10.1016/j.jnca.2016.03.016.

[69] M. Channegowda, R. Nejabati, and D. Simeonidou, "Software-defined optical networks technology and infrastructure: Enabling software-defined optical network operations," *J. Opt. Commun. Netw.*, vol. 5, no. 10, pp. A274–A282, Oct. 2013. [Online]. Available: http://jocn.osa.org/abstract.cfm?URI=jocn-5-10-A274

[70] P. Fröhlich, E. Gelenbe, and M. P. Nowak, "Smart SDN management of fog services," in *Proc. Global IoT Summit (GIoTS)*. Dublin, Ireland: IEEE Commun. Soc., Jun. 2020, pp. 1–6.

[71] F. Francois and E. Gelenbe, "Towards a cognitive routing engine for software defined networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[72] T. Czachorski, E. Gelenbe, G. S. Kuaban, and D. Marek, "Transient behaviour of a network router," in *Proc. 43rd Int. Conf. Telecommun. Signal Process. (TSP)*, Milan, Italy, Jul. 2020, pp. 1–5.

[73] F. Francois and E. Gelenbe, "Optimizing secure SDN-enabled inter-data centre overlay networks through cognitive routing," in *Proc. IEEE 24th Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst. (MASCOTS)*, Sep. 2016, pp. 283–288.

[74] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "QoS-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, San Francisco, CA, USA, Jun. 2016, pp. 25–33, doi: 10.1109/SCC.2016.12.

[75] C. Qiu, S. Cui, H. Yao, F. Xu, F. R. Yu, and C. Zhao, "A novel QoS-enabled load scheduling algorithm based on reinforcement learning in software-defined energy Internet," *Future Gener. Comput. Syst.*, vol. 92, pp. 43–51, Mar. 2019, doi: 10.1016/j.future.2018.09.023.

[76] E. Gelenbe, J. Domanska, T. Czachorski, A. Drosou, and D. Tzovaras, "Security for Internet of Things: The SerIoT project," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Rome, Italy, Jun. 2018, pp. 1–5, doi: 10.1109/ISNCC.2018.8531004.

[77] M. Nowak, S. Nowak, J. Domanska, and T. Czachorski, "Cognitive packet networks for the secure Internet of Things," in *Proc. Global IoT Summit*, Jun. 2019, pp. 1–4.

[78] L. Tello-Oquendo, S.-C. Lin, I. F. Akyildiz, and V. Pla, "Software-defined architecture for QoS-aware IoT deployments in 5G systems," *Ad Hoc Netw.*, vol. 93, Oct. 2019, Art. no. 101911, doi: 10.1016/j.adhoc.2019.101911.

[79] F. J. Hauck, E. Meier, U. Becker, M. Geier, U. Rastofer, and M. Steckermeier, "A middleware architecture for scalable, qos-aware, and self-organizing global services," in *Proc. 3rd Int. Working Conf., Trends Distrib. Syst., Towards Universal Service Market (USM IFIP/GI)* (Lecture Notes in Computer Science), vol. 1890, C. Linnhoff-Popien and H. Hegering, Eds. Munich, Germany: Springer, Sep. 2000, pp. 214–229, doi: 10.1007/10722515_18.

[80] C. K. Toh, *Ad Hoc Mobile Wireless Networks*. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[81] R. Gotzhein, *Real-Time Communication Protocols for Multi-Hop Ad-hoc Networks—Wireless Networking in Production and Control Systems* (Computer Communications and Networks). Berlin, Germany: Springer, 2020, doi: 10.1007/978-3-030-33319-5.

[82] E. Gelenbe, "Quality of service in ad hoc networks," *Ad Hoc Netw.*, vol. 2, no. 3, p. 203, Jul. 2004.

[83] Q. Liang, "Designing power aware self-reconfiguring topology for mobile wireless personal area networks using fuzzy logic," *IEEE Trans. Syst., Man Cybern. C, Appl. Rev.*, vol. 33, no. 3, pp. 390–394, Aug. 2003.

[84] E. Gelenbe and R. Lent, "Link quality-aware routing," in *Proc. 1st ACM Int. Workshop Perform. Eval. Wireless Ad Hoc, Sensor, Ubiquitous Netw. (PE-WASUN)*, M. Ould-Khaoua and F. Zambonelli, Eds., Venezia, Italy, Oct. 2004, pp. 87–90, doi: 10.1145/1023756.1023772.

[85] O. Brun, L. Wang, and E. Gelenbe, "Big data for autonomic intercontinental overlays," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 575–583, Mar. 2016.

[86] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of Internet path selection," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 289–299, Oct. 1999.

[87] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.

[88] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *Proc. Workshop Future Directions Netw. Archit. (ACM SIGCOMM)*, Portland, OR, USA, Aug./Sep. 2004, pp. 1–8. [Online]. Available: http://conferences.sigcomm.org/sigcomm/2004/workshop_papers/fdna01-feamster1.pdf

[89] E. Gelenbe, X. Mang, and R. Önvural, "Diffusion based statistical call admission control in ATM," *Perform. Eval.*, vols. 27–28, pp. 411–436, Oct. 1996.

[90] E. Gelenbe, "Réseaux neuronaux aléatoires stables," *Comptes Rendus de l'Académie des Science, Mécanique, Physique, Chimie, Sciences de l'Univers, Sciences de la Terre*, vol. 310, no. 3, pp. 177–180, 1990.

[91] S. Timotheou, "The random neural network: A survey," *Comput. J.*, vol. 53, no. 3, pp. 251–267, Mar. 2010.

[92] J. Snijders. (2019). *The NLNOG Ring*. [Online]. Available: https://ring.nlnog.net

[93] Amazon and EC2. (2019). *Ec2: Amazon Elastic Cloud*. [Online]. Available: https://www.amazonaws.cn/en/ec2/

[94] A. N. Tantawi and D. Towsley, "Optimal static load balancing in distributed computer systems," *J. ACM*, vol. 32, no. 2, pp. 445–465, Apr. 1985.

[95] C. Kim and H. Kameda, "An algorithm for optimal static load balancing in distributed computer systems," *IEEE Trans. Comput.*, vol. 41, no. 3, pp. 381–384, Mar. 1992.

[96] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[97] X. Zhu, X. Qin, and M. Qiu, "QoS-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters," *IEEE Trans. Comput.*, vol. 60, no. 6, pp. 800–812, Jun. 2011.

[98] W. Tian, Y. Zhao, Y. Zhong, M. Xu, and C. Jing, "A dynamic and integrated load-balancing scheduling algorithm for cloud datacenters," in *Proc. IEEE Int. Conf. Cloud Comput. Intell. Syst.*, Sep. 2011, pp. 311–315.

[99] Z. Zhang and X. Zhang, "A load balancing mechanism based on ant colony and complex

network theory in open cloud computing federation," in *Proc. 2nd Int. Conf. Ind. Mechatronics Autom.*, vol. 2, May 2010, pp. 240–243.

[100] E. Gelenbe and K. Sevcik, "Analysis of update synchronization for multiple copy data bases," in *Proc. 3rd Berkeley Workshop Distrib. Data Comput. Netw.*, 1978, pp. 69–90.

[101] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proc. 5th Int. Joint Conf. (IMS IDC)*, 2009, pp. 44–51.

[102] R. Buyya *et al.*, "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM Comput. Surveys*, vol. 51, no. 5, pp. 1–38, Jan. 2019.

[103] L. Wang, O. Brun, and E. Gelenbe, "Adaptive workload distribution for local and remote clouds," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Budapest, Hungary, Oct. 2016, pp. 3984–3988, doi: 10.1109/SMC.2016.7844856.

[104] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 33–45, Jan. 2018.

[105] H. Kameda, J. Li, C. Kim, and Y. Zhang, *Optimal Load Balancing in Distributed Computer Systems*. Berlin, Germany: Springer, 2011.

[106] E. Gelenbe, "Sensible decisions based on QoS," *Comput. Manage. Sci.*, vol. 1, no. 1, pp. 1–14, Dec. 2003.

[107] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Comput.*, vol. 1, no. 4, pp. 502–510, Dec. 1989.

[108] H. Abdelbaki, E. Gelenbe, and S. E. El-Khamy, "Analog hardware implementation of the random neural network model," in *Proc. IEEE Int. Joint Conf. Neural Netw. Neural Comput., New Challenges Perspect. New Millennium (INNS-ENNS)*, vol. 4, Jul. 2000, pp. 197–201.

[109] M. Badaroglu, U. Halici, I. Aybay, and C. Cerkez, "A cascadable random neural network chip with reconfigurable topology," *Comput. J.*, vol. 53, no. 3, pp. 289–303, Mar. 2010.

[110] E. Gelenbe, "Learning in the recurrent random neural network," *Neural Comput.*, vol. 5, no. 1, pp. 154–164, 1993.

[111] A. J. Jara, P. Lopez, D. Fernandez, M. A. Zamora, A. F. Skarmeta, and L. Marin, "Evaluation of Bluetooth low energy capabilities for tele-mobile monitoring in home-care," *J. Universal Comput. Sci.*, vol. 19, no. 9, pp. 1219–1241, 2013.

[112] E. Gelenbe and Y. M. Kadioglu, "Energy life-time of wireless nodes with network attacks and mitigation," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Kansas City, MO, USA, 2018, pp. 1–6, doi: 10.1109/ICCW.2018.8403561.

[113] K. Kalkan and S. Zeadally, "Securing Internet of Things with software defined networking," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 186–192, Sep. 2018.

[114] G. Sakellari, C. Morfopoulou, and E. Gelenbe, "Investigating the tradeoffs between power consumption and quality of service in a backbone network," *Future Internet*, vol. 5, no. 2, pp. 268–281, 2013.

[115] G. Oke, G. Loukas, and E. Gelenbe, "Detecting denial of service attacks with Bayesian classifiers and the random neural network," in *Proc. IEEE Int. Fuzzy Syst. Conf.*, London, U.K., Jun. 2007, pp. 1–6, doi: 10.1109/FUZZY.2007.4295666.

[116] (Mar. 2015). *OpenFlow Switch Specification, Version 1.3.5, Open Networking Foundation*. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.3.5.pdf

[117] K. Phemius and M. Bouet, "Monitoring latency with OpenFlow," in *Proc. 9th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2013, pp. 122–125.

[118] S. Salam, "Weight initialization for random neural network reinforcement learning," U.S. Patent Appl. US 2019 097 912 AI, Mar. 9, 2019.

[119] J. Du, C. Jiang, E. Gelenbe, L. Xu, J. Li, and Y. Ren, "Distributed data privacy preservation in IoT applications," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 68–76, Dec. 2018.

[120] J. Augusto-Gonzalez *et al.*, "From Internet of threats to Internet of Things: A cyber security architecture for smart homes," in *Proc. IEEE 24th Int. Workshop Comput. Aided Modeling Design Commun. Links Netw. (CAMAD)*, Limassol, Cyprus, Sep. 2019, pp. 1–6, doi: 10.1109/CAMAD.2019.8858493.

[121] P. Natsiavas *et al.*, "Comprehensive user requirements engineering methodology for secure and interoperable health data exchange," *BMC Med. Informat. Decis. Making*, vol. 18, no. 1, p. 85, Dec. 2018.

[122] O. Brun, Y. Yin, and E. Gelenbe, "Deep learning with dense random neural network for detecting attacks against IoT-connected home environments," *Procedia Comput. Sci.*, vol. 134, pp. 458–463, Jan. 2018.

[123] L. Delosieres and D. Garcia, "Infrastructure for detecting Android malware," in *Proc. 28th Int. Symp. Comput. Inf. Sci. (ISCIS)* (Lecture Notes in Electrical Engineering), vol. 264, E. Gelenbe, Ed. Cham, Switzerland: Springer, Oct. 2013.

[124] E. Gelenbe and A. Stafylopatis, "Global behavior of homogeneous random neural systems," *Appl. Math. Model.*, vol. 15, no. 10, pp. 534–541, Oct. 1991.

[125] L. Delosieres and A. Sanchez, "DroidCollector: A honeyclient for collecting and classifying Android applications," in *Proc. Inf. Sci. Syst., 29th Int. Symp. Comput. Inf. Sci. (ISCIS)*, T. Czachórski, E. Gelenbe, and R. Lent, Eds. Cham, Switzerland: Springer, Oct. 2014, pp. 175–183, doi: 10.1007/978-3-319-09465-6_19.

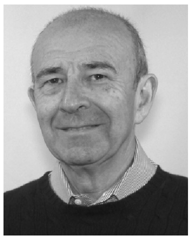
[126] U. Halici, "Reinforcement learning with internal expectation for the random neural network," *Eur. J. Oper. Res.*, vol. 126, no. 2, pp. 288–307, 2000, doi: 10.1016/S0377-2217(99)00479-8.

[127] A. Nguyen-Ngoc, S. Lange, S. Geissler, T. Zinner, and P. Tran-Gia, "Estimating the flow rule installation time of SDN switches when facing control plane delay," in *Proc. 19th Int. GI/ITG Conf., Meas., Modelling Eval. Comput. Syst. (MMB)* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2018, doi: 10.1007/978-3-319-74947-1.

[128] T. Czachorski, A. Domanski, J. Domanska, M. Pagano, and A. Rataj, "Delays in IP routers, a Markov model," in *Proc. Int. Symp. Comput. Inf. Sci. (ISCIS)*, vol. 659. Cham, Switzerland: Springer, 2016, pp. 185–192, doi: 10.1007/978-3-319-47217-1_25.

[129] A. Domanski, J. Domanska, M. Pagano, and T. Czachorski, "The fluid flow approximation of the TCP Vegas and Reno congestion control mechanism," in *Proc. Int. Symp. Comput. Inf. Sci. (ISCIS)*, vol. 659. Cham, Switzerland: Springer, 2016, pp. 193–200, doi: 10.1007/978-3-319-47217-1_25.

ABOUT THE AUTHORS



Erol Gelenbe (Life Fellow, IEEE) Fellow of the Association for Computing Machinery, the International Federation of Information Processing, the Royal Statistical Society, and the Institution of Engineering and Technology, was born in Istanbul, graduated from TED Ankara Koleji, and was awarded the BS (1966) by METU (Ankara), the MS (1968) and PhD (1970) by Polytechnic Institute of NYU, and the DSc (1973) degree by Sorbonne University, Paris. Professor at the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, his previous positions include Chaired Professorships at Imperial College, University of Central Florida, Duke University, Université Paris-Descartes and Paris-Orsay, and Université de Liège (Belgium). Renowned for pioneering mathematical models of computer systems and networks, inventing G-Networks and Random Neural Networks, he graduated over 90 PhDs, including 24 women. He has contributed to industry by developments that include the Queueing Network Analysis Package, the XANTOS fiber-optics local area network, the SYCOMORE distributed voice-packet switch, the manufacturing simulator FLEXSIM, and the C2 (Command and Control) Agents software. Awards include "honoris causa" degrees from Università di Roma II (1996), Bogazici University (Istanbul) (2004), and Université de Liège (Belgium) (2006), the Parlar Foundation Science Award (1994), Grand Prix France Télécom of the French Acad. of Sciences (1996), ACM-SIGMETRICS Life-Time Achievement Award (2008), IET Oliver Lodge Medal, (2010), and the "In Memoriam Dennis Gabor Prize" of the Hungarian Acad. of Sciences (2013). He was elected Fellow of Academia Europaea, the French National Acad. of Technologies, the Science Acad. of Turkey, the Royal Acad. of Belgium, the Hungarian and Polish Academies of Science. He was awarded the honors of Chevalier de la Légion d'Honneur, Chevalier des Palmes Académiques, and Commandeur du Mérite of France, and Commendatore al Merito and Grande Ufficiale dell'Ordine della Stella of Italy. He serves as associate Editor of IEEE TRANSACTIONS CLOUD COMPUTING, *Acta Informatica*, *Performance Evaluation*, and joint Editor-in-Chief of *SN Computer Science*, his consultancies have included INRIA, France-Telecom, Bull, IBM, Thomson CSF, Bell Labs, BT, General Dynamics UK, Huawei. Principal Investigator of many European Union research projects, Coordinator of FP7 NEMESYS and H2020 SerIoT, he has also won numerous grants from the National Science Foundation, the Engineering and Physical Sciences Research Council, industry and other organizations.

Joanna Domanska received the master's degree in engineering from the Silesian University of Technology, Gliwice, Poland, in 1994, and the Ph.D. degree in computer science from the Scientific Council of the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Gliwice, in January 2005, and defended the habilitation degree in computer science from the Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, in 2015.

She has taken part in seven projects for The National Science Centre, Kraków, Poland, and was the Project Leader in two of them. She is currently an Associate Professor with the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences (IITiS-PAN). She has been a member of the Computer Systems Modeling and Performance Evaluation Group, IITiS, Gliwice, Poland, since 1994 where she has been serving as the Leader since 2018. She is also a member of the SerIoT and the SDK4ED Research Teams through the EU H2020 Program. Her main areas of research interest include performance modeling methods for computer networks, particularly the modeling of network traffic intensity and the quality of service (QoS) related problems.

Piotr Fröhlich received the bachelor's degree in engineering from the Silesian University of Technology, Gliwice, Poland, in 2020, where he is currently pursuing the master's degree in engineering.

In 2018, he had already started research at the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences (IITiS-PAN), Gliwice, on network security and performance as a part of the EU H2020 Research and Innovation Project SerIoT. His field of interest revolves around artificial intelligence (AI), network security, and network efficiency, with an emphasis on the connections between these fields.

Mateusz P. Nowak was born in 1972. He received the M.Sc. degree in computer science from the Silesian University of Technology, Gliwice, Poland, in 1996, and the Ph.D. degree in computer networks with the Institute of Theoretical and Applied Computer Science, Polish Academy of Sciences (IITiS-PAN), Gliwice, in 2006.

He then worked as a software developer for Polish and German companies. Then, he joined the team on Systems Modeling and Performance Evaluation of Computer System, IITiS-PAN, as an Assistant Professor, and has participated as a researcher or as a principal investigator in several scientific and industry research and development projects. His research interests have covered issues related to distributed programming, event-driven simulation, and novel solutions for the future Internet. His current research interests include various aspects of devices and communication technologies in the Internet of Things (IoT), with an emphasis on network security.

Sławomir Nowak was born in Zabrze, Poland, in November 9, 1974. He received the M.Sc. degree in computer systems and networks from the Faculty of Computer Science, Silesian University of Technology, Gliwice, Poland, in 1999, and the Ph.D. degree in computer science from the Silesian University of Technology in 2005.

In 2000, he joined the System Modeling and Performance Evaluation of Computer Systems Group, Institute of Theoretical and Applied Informatics, Polish Academy of Science (IITiS-PAN), Gliwice. In 2005, he joined ITIS-PAN as an Assistant Professor. He is currently active in the EU H2020 Research and Innovation Project SerIoT. Since then, his research interests include discrete event simulation, sensor networks, the future Internet, and the Internet of Things (IoT). He has collaborated actively with commercial companies in research and development projects in many different areas regarding the application of computer networks and information systems.